all rights reversed

HACKER

Season 05 / Issue 05

τεύχος 053 - έκρηξη-ανάπτυξης Edition

Ανάλυση σοβαρής αδυναμίας της glibc [*]

τι είναι το game loop και η σπουδαιότητά του

πρώτη πνοή ζωής για το παιχνίδι που αναπτύσσουμε

πώς κάνουμε το chat *πραγματικά* προσωπική μας υπόθεση

επικοινωνία κάρτας ήχου-συστήματος, στον DIY υπολογιστή μας

[*] ...και επεξήγηση βασικών εννοιών

all rights reversed

HACKER


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

#dH053 | Περιεχόμενα

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Γιώργος Γιάννου, Καραγιαννόπουλος Μάριος, Γιάννης Κρομμύδας, Πέτρος Κυλαδίτης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας, @arkoudos
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στην Καλαμαριά Θεσσαλονίκης. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/pdf12order και από την ίδια σελίδα γίνονται οι παραγγελίες συνδρομών.


deltaHacker Field Bag
€25.50
MORE COLORS AVAILABLE
deltaHacker Mugs
€12.50
deltaHacker Thermos® Can Cooler
€17.50
QUESTIONS? | ORDER BY PHONE? | CALL 1-877-809-1659
Where's My Stuff?
Track your recent orders.
View or change your order.
Shipping & Returns
See our shipping rates.
Make a return or exchange.
Need Help?
Satisfaction Guaranteed.
Visit our Customer Service.
is powered by CafePress Inc. Copyright © 1999-2015 CafePress.com. All rights reserved.
olicy | Trademark & Copyright Information
urrency rates are updated daily and may fluctuate.
Δώρα με στιλ, που έχουν *αληθινή* αξία.
Όλα στο μαγαζάκι του deltaHacker,
κάτω από το Cafepress.
Περάστε μια βόλτα από το
http://www.cafepress.co.uk/deltahacker
–σας περιμένουμε!

Παραξενιές και Όπερα

του Χρήστου Βαρελά

Αγαπητές κι αγαπητοί,

- Η μέρα σήμερα, Κυριακή 24/4, είναι κάπως παράξενη.
- Κάθομαι τώρα, πρωί, και πληκτρολογώ τις πρώτες γραμμές του editorial για το τεύχος 053.
- Έχω την τηλεόραση για παρέα, με χαμηλωμένη την ένταση, σε ένα από τα Βοθροκάναλα της Διαπλοκής (TM). Παίζει αυτή την παλιά σειρά με τη Τζίνη, το τζίνι.
- Κανονικά θα είχα για παρέα άλλο Βοθροκάναλο της Διαπλοκής (TM), αλλά αυτές τις μέρες δεν έχει ενημερωτικές εκπομπές. Είναι που η ΕΣΗΕΑ αποφάσισε να ταράξει την κυβέρνηση στην απεργία. Μιλάμε τώρα για απεργία χωρίς προηγούμενο, η οποία αν δεν απατώμαι ήταν να τελειώσει σήμερα αλλά τελικά θα πάει έως τη Μεγάλη Τετάρτη. Κάτι τέτοιο είδα στο Twitter κι αυτό μου επαληθεύει και η κα Σύμβουλος Εκδόσεως (TM) του περιοδικού.
- Τέλος πάντων.
- Το τεύχος 053, που λέτε, σε σχέση με το προηγούμενο βγαίνει αρκετά πιο νωρίς. Αργότερα, προς το απογευματάκι, θα έχουν γίνει και οι τελευταίες διορθώσεις στο PDF και οι συνδρομητές θα λάβουν ενημερωτικό email, ώστε να πάνε στο site και να το κατεβάσουν και να το διαβάσουν και να ευχαριστηθούν (hopefully).
- Το ότι το νέο τεύχος βγαίνει νωρίς είναι ένα από τα παράξενα της ημέρας. Δεν το λες και δυσάρεστο.
- Ένα άλλο παράξενο, το οποίο επίσης δεν λες δυσάρεστο, είναι ότι από χθες έχουμε ωραία δροσούλα. Όλες οι μετρήσεις επαληθεύουν πως ο χειμώνας που μας πέρασε ήταν τέτοιος μόνο κατ' όνομα. Επίσης, για άλλη μια φορά οι επιστήμονες συμφωνούν πως η κλιματική αλλαγή οφείλεται στον άνθρωπο. Δεν πρόκειται για περιστασιακό φαινόμενο, εμείς το προκαλούμε, τα πράγματα αλλάζουν δραματικά και, ακόμη κι αν περιορίσουμε εντυπωσιακά τις εκπομπές αερίων του θερμοκηπίου, η αδράνεια που έχει το σύστημα "καιρός" είναι τόσο μεγάλη, ώστε η περαιτέρω άνοδος της μέσης θερμοκρασίας να είναι αναπόφευκτη.
- Μ' άλλα λόγια: Όλοι όσοι βλέπουν τη ζωή σαν μια απέραντη παραλία με έντονα χαρακτηριστικά Μυκόνου, έχουν τουλάχιστον ένα λόγο για να χαίρονται. Είναι που συνειδητοποιούν πως το όνειρό τους για "καλοκαιρινές διακοπές για πάντα" έχει καλές πιθανότητες να γίνει πραγματικότητα.
- Εντάξει, καταλαβαίνω ότι διαφορετικοί άνθρωποι έχουν διαφορετικά όνειρα, κατά την ταπεινή μου γνώμη όμως το "καλοκαιρινές διακοπές για πάντα" όνειρο είναι αρκετά παραπλανητικό.

- Εξηγούμαι: Το ότι το κλίμα αλλάζει κι όπως οι Σύντροφοι (TM) έτσι και η ζέστη "κερδάει", δεν σημαίνει ότι θα κάνουμε και περισσότερο διακοπές μέσα στο χρόνο.
- Ας έχουν οι οπαδοί της ζέστης κατά νου ότι όλα δείχνουν πως, στη χώρα μας τουλάχιστον, όσοι έχουμε κάποια δουλειά θα δουλεύουμε ολοένα και περισσότερο. Κι ενώ θα δουλεύουμε ολοένα και περισσότερο και θα κοπιάζουμε και θα τρέχουμε και δεν θα φτάνουμε, την ίδια στιγμή θα κάνει ζέστη. Πολλή ζέστη. Κάπως έτσι θα είναι η κλιματική αλλαγή για εμάς -- και μάλλον περιγράφω ένα αισιόδοξο σενάριο.
- Τέλος πάντων.
- Για να ελαφρύνω λίγο το κλίμα (no pun intended), αλλάζω απότομα θέμα. Κάνω στροφή 360 μοιρών, όπως είχε πει κι ο επικεφαλής των Σκληρών Διαπραγματευτών (TM), και στο καπάκι ρίχνω άλλη μία --μισή αυτή τη φορά-- στροφή.
- Θα τα μάθατε τα νέα για τον Opera browser, έτσι δεν είναι; Την περασμένη Πέμπτη ανακοινώθηκε ότι η πλέον πρόσφατη έκδοση της σειράς developer του browser, περιλαμβάνει λειτουργικότητα VPN. Μιλάμε μάλιστα για δωρεάν VPN, χωρίς περιορισμούς.
- Η υπόσχεση είναι ότι, πολύ εύκολα και χωρίς καμία προετοιμασία, χάρη στο VPN του Opera όποτε το επιθυμούμε θα μπορούμε να κρύβουμε την πραγματική διεύθυνση IP με την οποία βγαίνουμε στο ίντερνετ, να παρακάμπτουμε τους περιορισμούς που πιθανώς ισχύουν στο δίκτυο εντός του οποίου βρισκόμαστε, βεβαίως και να ενισχύουμε την ασφάλειά μας στα δημόσια δίκτυα WiFi, στα αεροδρόμια, στα ξενοδοχεία, στα βαπόρια κ.ο.κ.
- Προς το παρόν, στην developer έκδοση του Opera οι χρήστες επιλέγουν μεταξύ τριών γεωγραφικών περιοχών εξόδου στο "κανονικό" ίντερνετ: ΗΠΑ, Καναδάς και Γερμανία. Σε επόμενη, σταθερή έκδοση του Opera, θα διατίθενται περισσότερες γεωγραφικές περιοχές.
- Ωραίο δεν ακούγεται; Δωρεάν κι εύκολο VPN, ανά πάσα στιγμή και για όλους! Επόμενο ήταν να γίνει χαμός στα social τα media, με τα RT και τα Share και τα Like να πέφτουν βροχή.
- Εμείς πάντως, ως περιοδικό, κρατήσαμε μια πιο επιφυλακτική στάση. Την εκφράσαμε και στο Twitter, αν και κάτι μας λέει πως σημασία έδωσαν το πολύ οκτώ followers.
- Με την υπόθεση VPN έχουμε ασχοληθεί -- και μάλιστα αρκετοί συνδρομητές θα έλεγαν πως έχουμε παρασχοληθεί.
- Έχουμε, π.χ., δείξει, αναλυτικά και βήμα προς βήμα, πώς στήνουμε τον δικό μας OpenVPN server, στο σπίτι μας ή στο cloud. Κι επειδή το project απαιτεί μια κάποια τεχνογνωσία και χρειάζεται λίγη δουλειά, δείξαμε και πώς στήνουμε τον λεγόμενο Access Server, στο cloud. Μετά κάναμε κι ολόκληρο αφιέρωμα περί της προστασίας του privacy στο ίντερνετ. Στο πλαίσιο του αφιερώματος, επικεντρώσαμε την προσοχή μας στην επί πληρωμή υπηρεσία VPN του TorGuard. Χάρη σε μια τέτοια υπηρεσία έχουμε τη δυνατότητα επιλογής μεταξύ δεκά-

δων γεωγραφικών σημείων εξόδου ανά τον πλανήτη. Παρέχονται επίσης και τεχνικές "μασκαρέματος" του VPN traffic, ώστε ο ISP να μην είναι καν σε θέση να βλέπει ότι χρησιμοποιούμε VPN κι έτσι να έχει μια δικαιολογία λιγότερη για bandwidth throttling.

- Ανεξάρτητα του VPN server που στήνουμε ή της υπηρεσίας VPN που επιλέγουμε, το καλό είναι ότι όλη, μα όλη η δικτυακή κίνηση του υπολογιστή μας, αναδρομολογείται μέσω ενός απομακρυσμένου μηχανήματος. Μεταξύ του υπολογιστή μας και του προαναφερθέντος μηχανήματος, τα δεδομένα ταξιδεύουν μέσα από ένα ισχυρά κρυπτογραφημένο κανάλι.
- Το όχι-και-τόσο-καλό με το VPN της Opera, είναι ότι από νωρίς φάνηκε --χωρίς φυσικά να διαφημιστεί-- ότι μόνον η κίνηση του browser, όχι ολόκληρου του υπολογιστή, ταξιδεύει μέσω κρυπτογραφημένου καναλιού. Με άλλα λόγια, η προστασία που παρέχει ο VPN του Opera είναι ελλιπής: η πραγματική μας ταυτότητα διαρρέει στο ίντερνετ -- και μάλιστα διαρρέει με χίλιους δυο τρόπους.
- Μία μόλις μέρα μετά την ανακοίνωση της κατασκευάστριας του Opera, ερευνητές ασφαλείας επαλήθευσαν ότι η λειτουργικότητα VPN του browser συγγενεύει περισσότερο με proxy, παρά με αληθινό VPN. Ουσιαστικά, είναι σαν να επεμβαίνουμε στις ρυθμίσεις του browser και να του λέμε "κάνε χρήση αυτού του proxy". Στο εξής, ο Opera πράγματι θα στέλνει και θα λαμβάνει δικτυακά πακέτα με ένα μόνο απομακρυσμένο μηχάνημα (ο proxy), ενώ όταν επισκεπτόμαστε κάποιο site θα καταγράφεται το δημόσιο IP του proxy κι όχι του υπολογιστή μας. Όλες όμως οι άλλες εφαρμογές μας, βεβαίως και το ίδιο το λειτουργικό, δεν θα κάνουν χρήση του εν λόγω proxy. Σε αντίθεση λοιπόν με μια αληθινή υπηρεσία VPN που είναι system wide, ο Opera προσφέρει μια ψευδαίσθηση ασφάλειας.
- Παράξενο;
- Όχι και τόσο, αν σκεφτούμε πως μιλάμε για έναν απλό browser που διατίθεται για κάθε λειτουργικό, και οι developers δεν θέλουν να μπλέξουν με αλλαγές στις παραμέτρους δικτύωσης του εκάστοτε συστήματος.
- Αλλά τι να κάνουμε, το μάρκετινγκ είναι μάρκετινγκ και οι υπεύθυνοι της εταιρείας αποφάσισαν να τραβήξουν την προσοχή μας στη λειτουργικότητα VPN.
- Τέλος πάντων.
- Σας αφήνουμε να χαρείτε το νέο τεύχος και να γιορτάσετε την Ανάσταση του Θεανθρώπου, βεβαίως και της οικονομίας.
- Η τελευταία έχει ανακοινωθεί κι από επίσημες κυβερνητικές πηγές και είμαστε βέβαιοι ότι δεν πρόκειται για κίνηση μάρκετινγκ.
- Κι όσο για τη χρονική σύμπτωση των δύο αναστάσεων, προφανώς και πρόκειται για άλλη μια (χαριτωμένη) παραξενιά των ημερών.

POST CARD

THE ADDRESS TO BE WRITTEN ON THIS SIDE



Ονέιλ μέιλ

Αγαπητοί μου,

Λοιπόν, θα σας πω πώς το είχα σκεφτεί γι' αυτόν τον μήνα. (Είναι Απρίλιος, θα το έχετε προσέξει, δεν μπορεί)

Λοιπόν. Απρίλιος. Πρωταπριλιά, you know.

Αρχικά είχα σκεφτεί να σας γράψω κάτι Πρωταπριλιάτικο. Κάτι θα έβρισκα, θα το έγραφα, θα το "πιστεύατε", θα χαμογελάγαμε όλοι μαζί, και μετά θα συνεχίζαμε χαμογελαστά ο καθένας με τις δουλειές του.

Το ζύγισα στο μυαλό μου, το σκέφτηκα από εδώ κι από εκεί, και τελικά αποφάσισα να μην το κάνω, κυρίως γιατί οι επιστολές αυτές έχουν χρόνο ημισείας ζωής που ξεπερνά τη μία ημέρα, οπότε κάποιος μπορεί να τη διάβαζε στις 2 ή στις 3 Απριλίου, οπότε θα χανόταν και το στοιχείο της έκπληξης και η χαριτωμενιά του Πρωταπριλιάτικου.

Μετά σκέφτηκα να κάνω το αντίθετο.

Σκεφτόμουν να διατρέξω τις ειδήσεις της ημέρας (ή της εβδομάδας) και να συγκέντρωνα στην επιστολή μου κάποιες που έχουν πρωταπριλιάτικο ενδιαφέρον, ακόμα και αν δεν ήταν πρωταπριλιάτικες.

Για παράδειγμα:

"Λεκτικό λάθος χαρακτήρισε την επίμαχη αναφορά των Σκοπίων ως 'Μακεδονία' του αναπληρωτή υπουργού μεταναστευτικής πολιτικής, Γιάννη Μουζάλα, στον ΣΚΑΪ, ο υπουργός Εθνικής Άμυνας και επικεφαλής των ΑΝΕΛ, Πάνος Καμμένος, κατά τη διάρκεια της ομιλίας του στην προ ημερησίας διατάξεως συζήτηση για τη Δικαιοσύνη."

Who would think that, που λένε και οι κουτόφραγκοι.

Ή άλλο παράδειγμα:

"Απολαυστικό διάλογο μπροστά στις κάμερες και τους φωτορεπόρτερ είχαν ο Πρωθυπουργός Αλ. Τσίπρας με τον νέο επικεφαλής του Ελληνικού Φεστιβάλ, Γιαν Φαμπρ. 'Βασιζόμαστε πολύ σε εσάς...', είπε στον Βέλγο καλλιτέχνη ο Έλληνας πρωθυπουργός. 'Μη βασίζεστε και τόσο πολύ...', του απάντησε ο Γ. Φαμπρ γελώντας."

Ή τέλος:

"Κατά την τελετή αναγόρευσής του σε επίτιμο διδάκτορα του Εβραϊκού Πανεπιστημίου της Ιερουσαλήμ, ο πρόεδρος της Δημοκρατίας Πρ. Παυλόπουλος έκανε την ακόλουθη δήλωση: 'Μέσα σε αυτόν τον οικονομικό Λαβύρινθο και αντιμέτωπος με τον νεοφιλελεύθερο Μινώταυρο, ο θεσμικός Θησέας καλείται, υπό όρους που παραπέμπουν στην ιστορία του Δαυίδ απέναντι στον Γολιάθ, να φέρει σε πέρας την αποστολή του.'"

Βάζω μόνο αυτά τα τρία παραδείγματα, αγαπητοί μου, από τα δεκάδες που εντόπισα, για να σας δείξω πόσο εύκολη θα είχε γίνει η δουλειά μου αν επέτρεπα στον εαυτό μου τέτοια φτηνά τεχνάσματα.

Διότι, με την κατεύθυνση που έχουν πάρει τα δημόσια πράγματα στη χώρα μας, το να κοιτάς για να εντοπίσεις κάτι φαιδρό στην ειδησεογραφία είναι σαν να ληστεύεις εκκλησία, που λέει και το ρητό.

Στην πραγματικότητα η δυσκολία πλέον είναι να βρεις και να αναδείξεις κάτι το οποίο αντιμετωπίζεται και παρουσιάζεται με σοβαρότητα.

Ακόμα και κρισιμότατα ζητήματα, ζητήματα που άπτονται των ανθρωπίνων δικαιωμάτων, της αξιοπιστίας της χώρας, του ίδιου του μέλλοντός μας, αντιμετωπίζονται σήμερα με όρους καφενείου ή ουζάδικου, που πάμε με τα κολλητάρια και λέμε τι θα κάναμε αν ήμασταν κυβέρνηση.

Παραδείγματα; Να σας βρω παραδείγματα.

Βγαίνει ο υπουργός κύριος τάδε και μιλάει για την ευνοϊκή μεταχείριση που θα έχουν οι πρόσφυγες οι οποίοι είναι διατεθειμένοι να μεταφέρουν στην Ελλάδα χρήματα άνω των 250 χιλιάδων ευρώ, που ενδεχομένως έχουν σε ξένες τράπεζες.

Το λέει στα σοβαρά. Δεν το μετανιώνει. Το έψαξε στο Google, μας εξηγεί. Υπάρχει διεθνής εμπειρία.

Αστεία πράγματα. Διεθνής εμπειρία. Google.

Αλλά το πιο αστείο δεν είναι εδώ.

Το πιο αστείο είναι πως μετά την επιμονή του στελέχους της κυβέρνησης, που φωνάζει πως μιλάει σοβαρά και πως το έχει ψάξει το θέμα, κ.λπ. κ.λπ., το ίδιο το θέμα ξεχνιέται. Πάει στην άκρη. Τσιμουδιά. Μόκο. Λες κι ήταν ψες.

Κάτι σαν τις ιδέες εκείνου του άλλου στελέχους της κυβέρνησης --*πρώην* στελέχους για να είμαστε ακριβείς-- που έλεγε για τους καλωδιωμένους τουρίστες που θα καταπολεμάνε τη φοροδιαφυγή.

Η της άλλης, της λαοπρόβλητης βουλευτού --*πρώην* βουλευτού-- που έλεγε πως αν μας κόψουν τα δανεικά θα τυπώσουμε πεντακοσάρικα. Το έλεγε και το εννοούσε. Μην κοιτάτε που δεν είναι πλέον βουλευτής. Το εννοούσε.

Άλλο παράδειγμα σοβαρότητας, πρόσφατο αυτή τη φορά:

Βγαίνει το Wikileaks και παρουσιάζει υποκλοπή διαλόγου μεταξύ υψηλών παραγόντων του ΔΝΤ, σχετικά με την εκκρεμούσα ακόμα αξιολόγηση του Ελληνικού προγράμματος, ο οποίος (υποκλαπείς) διάλογος αποτελεί κατά την άποψη κάποιων σοβαρών υποτίθεται εφημερίδων "βόμβα" του ΔΝΤ για τη διαπραγμάτευση.

Ετοιμοπόλεμος, ο Πρωθυπουργός της Γαλανής, γνωστός και ως Aleksis (TM), καλεί τους συνεργάτες του σε συμβούλιο για να αποφασίσουν πώς θ' αντιδράσουν.

Το ψάχνουν από εδώ, το ψάχνουν από εκεί και τελικά ο Aleksis στέλνει επιστολή στο Χριστινάκι, όπου μιλάει για "κλονισμό της εμπιστοσύνης".

Ψώνιο.

Το άλλο, με τον Τοτό, το ξέρετε;

Ας μην ξεφεύγω. Το πράγμα είναι σοβαρό. Η εμπιστοσύνη κλονίζεται. Το ΔΝΤ βάζει βόμβες. Η διαπραγμάτευση είναι στον αέρα. Και λοιπά.

Δεν ξέρω εσείς, αλλά εγώ τουλάχιστον έκανα έναν κόπο. Βγήκα το αγγλικό κείμενο της διαρροής στο Wikileaks και το διάβασα ολόκληρο.

Πέραν των αναφορών σε τρέχοντα ζητήματα (Προσφυγικό, Brexit, κ.λπ.) δεν βρήκα τίποτα στο συγκεκριμένο κείμενο που να μην ήταν ήδη γνωστό.

Θα θυμάστε, αγαπητοί μου, σας έχω ξαναμιλήσει για το Χριστινάκι και την θέση του ΔΝΤ. Ρίξτε μια ματιά. Ακάκιοςleaks: επιστολή 044, https://deltahacker.gr/snailmail044.

Πείτε μου. Τι από αυτά που σας έγραφα εγώ τότε είναι τόσο διαφορετικό --με την έννοια του ασύμβατο, αντιφατικό-- από αυτά που διαρρέουν αυτές τις μέρες;

Δεν το ξέραμε πως για το ΔΝΤ το Ελληνικό πρόγραμμα δεν βγαίνει αν δεν μειωθεί το χρέος; Δεν το ξέραμε πως οι άλλοι δεν το συζητούν καν να μειωθεί το χρέος, αν δεν προχωρήσουν "οι μεταρρυθμίσεις"; Δεν το ξέραμε πως αν είναι να αποχωρήσει το ΔΝΤ, θα πρέπει να του επιστραφούν όσα έχει δώσει; Δεν το ξέραμε πως αποκλείεται να του επιστραφούν, γιατί αυτό θα σήμαινε πως οι



Ευρωπαίοι θα πρέπει να τα βάλουν από την τσέπη τους; Δεν το ξέραμε πως εφόσον το ΔΝΤ είναι σχεδόν εξαναγκασμένο να παραμείνει στο παιχνίδι, θα πρέπει η Ελλάδα να πάρει και άλλα μέτρα; Δεν το ξέραμε πως αυτό κάνει, ότι δηλαδή πιέζει για μέτρα;

Όλα αυτά τα ξέραμε. *Εγώ* τα ήξερα. Σας τα είχα γράψει. Άρα τα ξέρατε κι εσείς. Γιατί απορούμε, λοιπόν; Πού ακριβώς βρίσκεται η βόμβα; Πώς κλονίζεται η εμπιστοσύνη; Τι εννοεί ο Aleksis; Εννοεί πράγματι κάτι ο Aleksis;

Και, τέλος πάντων, ποιος επιτέλους είναι αυτός ο Aleksis; Υπάρχει πράγματι ή απλώς προήδρευε;

Αυτό εννοούσα, αγαπητοί μου, όταν έλεγα πως κρισιμότατα ζητήματα αντιμετωπίζονται με αγωνιστικό πνεύμα συντροφικής παρέας σε ουζάδικο.

Το χειρότερο είναι πως όταν κάνεις τον κόπο να το επισημάνεις, σου την πέφτουν όλοι με τα αριστερόμετρα και σε βγάζουν αφελή, νεοφιλελέ, χίπστερ, ηλίθιο, και δεν ξέρω τι άλλο.

Τέλος πάντων.

Διάβαζα αυτά που έγραφε για τους υποτιθέμενους κινδύνους από την Τεχνητή Νοημοσύνη ο Κύριος Εκδότης στο πιο πρόσφατο εντιτόριάλ του, και σκεφτόμουν πως ο μεγαλύτερος --και απολύτως πραγματικός-- κίνδυνος που αντιμετωπίζουμε στην Γαλανή μας Πατρίδα, είναι άλλος.

Να το πω αλλιώς: Μακάρι να είχαμε κι εμείς λίγη Νοημοσύνη, αγαπητοί μου, κι ας ήτανε και Τεχνητή. Αμ δε. Ο μεγαλύτερος κίνδυνος που αντιμετωπίζει η χώρα μας είναι αυτός της ολοκληρωτικής αποχαύνωσης λόγω συλλογικής βλακείας.

Αυτά σας τα είχα γράψει ήδη προ τριετίας, στην επιστολή 018 (https://deltahacker.gr/snailmail018). Πριν μας προκύψουν Aleksis και λοιποί. (Που, στο κάτω κάτω, εμείς τους βγάλαμε.)

"Η βλακεία σε μια κοινωνία", έγραφα τον Μάρτιο του '13, "είναι σαν τη γρίπη. Είναι μόλυνση. Ιωση. Οι ασπιρίνες και τα μαντζούνια δεν βοηθάνε. Πρέπει να περιμένεις να κάνει τον κύκλο της."

Και κατέληγα:

"Αν με ρωτήσετε πόσο καιρό θέλει ο κύκλος της συγκεκριμένης γρίπης, θα σας γελάσω. Αλλά δεν είμαι αισιόδοξος. Θα χρειαστούμε καμιά τριανταριά χρόνια, χαλαρά. Να μην πω και παραπάνω. Γιατί καμιά φορά από γρίπη παθαίνεις και πνευμονία."

Αυτά προ τριετίας.

Κοίταγα τώρα στον ιντερνέ, και παίρνει το μάτι μου την εξής είδηση:

"Με μαθηματικό κώδικα έγραψε ο Όμηρος τα έπη. 'Πυρήνας' είναι η πυθαγόρεια αρμονία και αριθμός-κλειδί το 3."

"Ό,τι και να κάνει, ο Όμηρος σχηματίζει πάντα και με επιμονή μαθηματικό λόγο", εξηγεί κάποια κυρία τάδε, διδάκτωρ φιλοσοφίας. "Για να δώσει μαθηματικές προεκτάσεις στους στίχους του, ο Όμηρος χρησιμοποιεί", λέει, "τους λεξάριθμους, στους οποίους το γράμμα 'άλφα' αντιστοιχεί στον αριθμό 'ένα', το 'Β' στο δύο και ούτω καθεξής."

Ημερομηνία της είδησης, η 1η Απριλίου. Οι αναγνώστες του ιντερνέ την εκλαμβάνουν ως πρωταπριλιάτικη. Το γράφουν στα σχόλια. Γελάνε.

Αλλά η είδηση δεν είναι πρωταπριλιάτικη. Η κυρία τάδε είναι όντως διδάκτωρ φιλοσοφίας και έκανε όντως αυτή την ανακοίνωση. Στην 8η Μαθηματική Εβδομάδα 2016. Της Ελληνικής Μαθηματικής Εταιρείας.

Βλέπετε τι εννοώ;

I rest my case.

Σας ασπάζομαι,
Θείος Ακάκιος



e-card
4000 1234 5678 9010
4000
GOOD THRU
MONTH / YEAR
12 / 19
MasterCard
Cardholder Name
Κάρτα e-coin, που δέχεται Bitcoins: Δουλεύει όπως όλες οι κανονικές κάρτες Debit, παντού στον κόσμο και για online αγορές.
Αποκτήστε τη δική σας με κλικ εδώ
και κερδίστε αυτομάτως
25% έκπτωση
κατά την παραγγελία της κάρτας.

Skill: Intermediate
Tags: glibc, remote code execution, stack, heap, LIFO, stack overflow, proof of concept, vuln

Ανάλυση κρίσιμου προβλήματος ασφαλείας στη glibc

Αναρωτηθήκατε ποτέ αν το λειτουργικό σύστημα που τρέχει στον υπολογιστή σας ή σε αυτόν κάποιου πελάτη, έχει τρύπες ασφάλειας; Κάτι μας λέει ότι, αν κι αυτή η έννοια είναι γνωστή σε όλους, σχεδόν κανένας δεν ασχολείται σοβαρά. Ίσως επειδή όλοι πιστεύουμε πως, ακόμη και να έχουμε κάποιο θέμα, στην επόμενη ενημέρωση του συστήματος τα πάντα θα έχουν διορθωθεί.

του Μάριου Καραγιαννόπουλου

Σωστή λογική, μόνο που κάποια κενά στην ασφάλεια είναι κρίσιμα και παρ' όλα αυτά αντιμετωπίζονται με μεγάλη καθυστέρηση. Τελικά, τα μηχανήματά μας είναι περισσότερο ευάλωτα απ' όσο νομίζουμε. Σ' αυτό το άρθρο θα εξετάσουμε μια κρίσιμη αδυναμία που δημοσιεύθηκε πρόσφατα.

Η ενασχόλησή μας με το θέμα ξεκίνησε ένα ήσυχο πρωινό, όταν ο γράφων άνοιξε το email στη δουλειά και βρήκε μια ντουζίνα ερωτήσεων κι απαντήσεων, γύρω από ένα ζήτημα ασφαλείας. Ένας από τους professional services architects της εταιρείας είχε διαβάσει ένα άκρως ανησυχητικό άρθρο και είχε πανικοβληθεί (http://www.theregister.co.uk/2016/02/16/glibc_linux_dns_vulernability). Με το δίκιο του ο άνθρωπος, αφού η αδυναμία για την οποία έκανε λόγο το άρθρο αφορούσε σε κάθε εφαρμογή που κάνει χρήση του DNS και βασίζεται στη βιβλιοθήκη glibc. Όλα τα σχετικά emails, λοιπόν, απευθύνονταν στους μηχανικούς που είχαν υλοποιήσει τέτοιες εφαρμογές και τους προέτρεπαν να τσεκάρουν κατά πόσο το bug επηρέαζε τα δικά τους προγράμματα.

Βάσεις αδυναμιών

Κατά πάσα πιθανότητα, κάθε μεγάλη εταιρεία που αναπτύσσει εφαρμογές ή μια ολόκληρη διανομή Linux, θα συντηρεί και κάποιο σύστημα για την έγκαιρη αντιμετώπιση των κενών ασφαλείας. Αυτός είναι ο λόγος που όταν δημοσιεύεται κάποια αδυναμία, περιγράφεται με κάποιο περίεργο ακρωνύμιο και έναν αριθμό. Για παράδειγμα, πολύ συχνά συναντάμε αυτά τα μυστήρια CVE και RHSA. Ας δούμε τι σημαίνουν.

- **CVE,** Common Vulnerability Exposure. Το CVE project (https://cve.mitre.org) συντηρείται από το MITRE Corporation, έναν μη κερδοσκοπικό οργανισμό που εκτελεί εργασίες έρευνας κι ανάπτυξης χρηματοδοτούμενες από την κυβέρνηση των ΗΠΑ.
- **RHSA,** RedHat Security Advisories. Συμβουλευτικά Δελτία Ασφαλείας από τη RedHat (https://access.redhat.com/security/security-updates/security-advisories). Η εταιρεία διατηρεί μια πολύ οργανωμένη βάση δεδομένων στην οποία συνδέει τα διάφορα CVEs με τα προϊόντα της. Το CVE για τη glibc βρίσκεται στο https://access.redhat.com/security/security-updates/security-advisories. Εκεί εξηγείται ποιο ακριβώς είναι το πρόβλημα και πώς λύνεται στα προγράμματα της εταιρείας. Επιπροσθέτως, αναφέρει αναλυτικά τα πακέτα που πρέπει να ενημερωθούν άμεσα.

Η αδυναμία ανακαλύφθηκε από έναν μηχανικό της Google, ο οποίος παρατήρησε ότι ο SSH client "έσκαγε" κάθε φορά που προσπαθούσε να συνδεθεί σ' ένα συγκεκριμένο σύστημα. Ο ίδιος μηχανικός καταχώρησε ένα ticket στο σχετικό σύστημα της εταιρείας και ξεκίνησε να διερευνά το πρόβλημα. Αυτό μπορεί να σας φαίνεται απίστευτο, αφού στις ελληνικές εταιρίες λογισμικού είναι αδιανόητο ότι κάποιος μηχανικός θα ασχοληθεί έστω και πέντε λεπτά με κάποιο θέμα που αφορά την παγκόσμια κοινότητα. Στις εταιρίες του Silicon Valley όμως τα πράγματα δεν είναι έτσι. Ειδικά στην Google, το 20% του χρόνου των μηχανικών αφιερώνεται σε projects ανοικτού λογισμικού. Τελικά, μετά από ενδελεχή έρευνα, οι μηχανικοί της Google που ασχολήθηκαν με το συγκεκριμένο θέμα ανακάλυψαν ότι το πρόβλημα βρίσκεται στη glibc κι όχι στο SSH. Μιλάμε για μια βασική βιβλιοθήκη (GNU libc) του Linux, που χρησιμοποιείται σε αμέτρητα προγράμματα. Η συγκεκριμένη ευπάθεια υπό συνθήκες επέτρεπε την απομακρυσμένη εκτέλεση κώδικα (remote code execution) στο σύστημα με την ευπαθή εφαρμογή. Περιττό να πούμε ότι μετά τον αρχικό εντοπισμό ξεκίνησε μια λεπτομερής ανάλυση του προβλήματος κι άρχισαν να γράφονται οι πιθανές λύσεις. Οι μηχανικοί της Google αντιμετώπισαν την όλη προσπάθεια σαν μια σπουδαία πρόκληση, καθώς η ευπάθεια επηρέαζε ένα τεράστιο πλήθος συστημάτων. Ξέρετε ποιο ήταν ένα από τα πρώτα πράγματα που ανακάλυψαν; Οι μηχανικοί που συντηρούσαν την glibc είχαν εντοπίσει την ευπάθεια κι είχαν καταχωρήσει ένα σχετικό bug από τον Ιούλιο του 2015! (https://sourceware.org/bugzilla/show_bug.cgi?id=18665) Συνεχίζοντας την έρευνα, οι μηχανικοί της Google διαπίστωσαν ότι το πρόβλημα είχε εντοπιστεί κι από δύο μηχανικούς της Red Hat, οι οποίοι είχαν αρχίσει να ερευνούν το θέμα ανεξάρτητα. Κάπως έτσι οι δύο ομάδες ξεκίνησαν τη συνεργασία και κατέληξαν στην έκδοση μιας αναλυτικής έκθεσης, που περιέγραφε το πρόβλημα και την αντιμετώπιση (δείτε το https://sourceware.org/ml/libc-alpha/2016-02/msg00416.html).

Το πρόβλημα

Οι αρχικές δοκιμές έδειξαν ότι η ευπάθεια υπήρχε σε όλες τις εκδόσεις της glibc, από τη 2.9 και μετά. Ένα τμήμα του κώδικα της βιβλιοθήκης, που σχετίζεται με το χειρισμό του DNS, είναι ευάλωτο και επιτρέπει την πραγματοποίηση επιθέσεων μέσω της υπερχείλισης στοίβας (stack overflow). Συγκεκριμένα, ο τρωτός κώδικας

βρίσκεται στη συνάρτηση getaddrinfo(). Τι σημαίνει αυτό στην πράξη; Τα προγράμματα που χρησιμοποιούν την εν λόγω συνάρτηση και απευθύνουν DNS queries προς διάφορους name servers, θα μπορούσαν να πέσουν θύματα ενός χάκερ που ελέγχει έναν τέτοιο server. Η ευπάθεια σχετίζεται με το μέγεθος της απάντησης (response) στο ερώτημα και εκδηλώνεται ανεξάρτητα από το αν χρησιμοποιείται το πρωτόκολλο UDP ή το TCP. Αν η απάντηση ξεπερνά σε μέγεθος τα 2048 bytes, προκαλείται υπερχείλιση στοίβας! Η προτεινόμενη λύση είναι ο περιορισμός του μεγέθους της απάντησης, είτε με το DNSMasq είτε με άλλα παρεμφερή προγράμματα. Επιπρόσθετα, πρέπει να φροντίζουμε ώστε τα ερωτήματα DNS να στέλνονται αποκλειστικά σε DNS servers που περιορίζουν το μέγεθος του response και χρησιμοποιούν και το truncation bit.

Αναγκαία ορολογία

Εσείς γνωρίζετε τους όρους stack και heap; Ο πρώτος μεταφράζεται σαν στοίβα και ο δεύτερος σαν σωρός, ενώ αναφέρονται και οι δύο σε περιοχές της μνήμης. Ωστόσο, η ακριβής ερμηνεία του καθενός δεν είναι γνωστή σε όλους. Για να κατανοήσουμε σε βάθος την ευπάθεια στη glibc, πρέπει να ξεδιαλύνουμε τους δύο όρους.

Με τον όρο stack αναφερόμαστε σε μια ειδική περιοχή μνήμης, όπου αποθηκεύονται οι μεταβλητές κάθε συνάρτησης. Κι όταν λέμε κάθε συνάρτηση, αναφερόμαστε ακόμα και στη main(). Η στοίβα αποτελεί μια δομή δεδομένων τύπου LIFO (Last In, First Out), την οποία διαχειρίζεται το λειτουργικό και ο επεξεργαστής. Κάθε φορά που ξεκινά η εκτέλεση μιας συνάρτησης, δεσμεύεται χώρος για τη δημιουργία μιας αντίστοιχης στοίβας. Η στοίβα μπορεί να μην έχει σταθερό μέγεθος, αλλά εκτείνεται σε μια περιοχή που έχει συγκεκριμένη έκταση. Στη στοίβα μιας συνάρτησης τοποθετούνται οι μεταβλητές που δημιουργούνται εντός της συγκεκριμένης συνάρτησης. Αντίστοιχα, όταν τερματίζεται μια συνάρτηση, η περιοχή της μνήμης που είχε δεσμευτεί για τη στοίβα απελευθερώνεται κι αυτό σημαίνει ότι όλες οι μεταβλητές της συνάρτησης διαγράφονται/χάνονται. Η χρήση της stack για την αποθήκευση μεταβλητών προσφέρει αρκετά πλεονεκτήματα. Το σημαντικότερο είναι ότι ο εκάστοτε προγραμματιστής δεν χρειάζεται να εκτιμήσει τον όγκο των δεδομένων και να δεσμεύει το αντίστοιχο ποσό μνήμης, ούτε να θυμηθεί να αποδεσμεύσει το σχετικό χώρο όταν δεν θα χρειάζεται τη μεταβλητή. Εξάλλου, η εγγραφή και η ανάγνωση μεταβλητών στη στοίβα πραγματοποιείται ταχύτατα. Τελικά, αυτά που πρέπει να θυμάται κανείς για τη στοίβα συνοψίζονται στα εξής:

- Η στοίβα μεγαλώνει καθώς μια συνάρτηση ορίζει νέες μεταβλητές.
- Η στοίβα μπορεί να επεκταθεί μέχρι ενός ορίου, αφού εκτείνεται σε μια περιοχή συγκεκριμένου μεγέθους.
- Ο προγραμματιστής δεν χρειάζεται να μπλέκει με τη διαχείριση μνήμης και αυτή η δουλειά πραγματοποιείται από το λειτουργικό σύστημα.
- Οι μεταβλητές της στοίβας υπάρχουν μόνο όσο για όσο εκτελείται η συνάρτηση που τις δημιούργησε.

Με τον όρο heap αναφερόμαστε σ' ένα κομμάτι μνήμης που διαχειρίζεται αποκλειστικά ο προγραμματιστής. Αυτή η περιοχή είναι αρκετά μεγαλύτερη από τη στοίβα. Στη γλώσσα C, για να δεσμεύσουμε ένα τμήμα του σωρού μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις malloc() και calloc(). Αφού δεσμεύσουμε μνήμη μέσα στο heap και τη χρησιμοποιήσουμε για όσο απαιτεί η εφαρμογή μας, πρέπει να θυμηθούμε να την αποδεσμεύσουμε. Κάτι τέτοιο, στη γλώσσα C πάντα, επιτυγχάνεται με τη συνάρτηση free(). Αν παραλείψουμε αυτό το στάδιο, καθώς το πρόγραμμά μας θα συνεχίζει να εκτελείται, η διαθέσιμη μνήμη στο σωρό θα εξαντλείται. Τότε λέμε ότι το πρόγραμμα παρουσιάζει διαρροή μνήμης (memory leak). Σε αντίθεση με τη stack, το heap δεν έχει περιορισμούς στο μέγεθος των μεταβλητών. Η πρόσβαση στο σωρό είναι πιο αργή απ' ό,τι στη στοίβα, αφού ο προγραμματιστής είναι υποχρεωμένος να χρησιμοποιεί δείκτες για να προσπελάσει τα αποθηκευμένα δεδομένα. Επίσης, σε αντίθεση με τις μεταβλητές που βρίσκονται στη στοίβα, οι μεταβλητές που δημιουργούνται στο heap είναι προσπελάσιμες από όλες τις συναρτήσεις του προγράμματος – είναι καθολικής έκτασης. Συνοψίζοντας, για το σωρό πρέπει να έχουμε υπόψη τα εξής:

- Η διαχείριση του σωρού αποτελεί ευθύνη του προγραμματιστή.
- Ο σωρός είναι μεγαλύτερος από τη στοίβα, αλλά ενδέχεται να σπαταλείται άσκοπα, όταν τα τμήματα που δεσμεύονται δεν απελευθερώνονται ποτέ.
- Οι μεταβλητές που τοποθετούνται στο heap είναι προσβάσιμες απ' όλο το πρόγραμμα.
- Σε γενικές γραμμές, δεν υπάρχει όριο στην έκταση του σωρού.

Περισσότερα είδη μνήμης

Το λειτουργικό σύστημα δεν δεσμεύει χώρο μόνο για τη στοίβα και το σωρό. Η έκταση της μνήμης που καταλαμβάνει κάθε εφαρμογή είναι μεγαλύτερη και χωρίζεται σε τμήματα που ονομάζονται segments. Αναλυτικότερα, έχουμε:

- **Code segment.** Σε αυτή την περιοχή της μνήμης τοποθετείται ο εκτελέσιμος κώδικας του προγράμματος. Συνήθως, το code segment επιτρέπει μόνο την ανάγνωση (read-only). Είναι γνωστό και σαν text segment.
- **Data segment.** Εκεί αποθηκεύονται οι μεταβλητές που έχουν δηλωθεί σαν global και οι οποίες είναι αρχικοποιημένες. Με άλλα λόγια, οι μεταβλητές που δηλώνονται στον κύριο κορμό του προγράμματος (έξω από οποιαδήποτε συνάρτηση), για τις οποίες έχουμε προσδιορίσει μια αρχική τιμή. Γι' αυτό το λόγο, το συγκεκριμένο τμήμα μνήμης ονομάζεται και initialized data segment.
- **BSS segment.** Εδώ τοποθετούνται οι μεταβλητές global, για τις οποίες δεν έχουμε ορίσει αρχική τιμή. Η συγκεκριμένη περιοχή ονομάζεται και uninitialized data segment.

Χειρισμός στην πράξη

Όταν στη C++ χρησιμοποιούμε τον τελεστή new για να δεσμεύσουμε μνήμη ο σχετικός χώρος καταλαμβάνεται από το σωρό (heap segment), που είναι γνωστός και σαν "free store".

```
int *ptr = new int;
int *array = new int[10];
```

Στην πρώτη γραμμή δηλώνουμε έναν δείκτη προς ακεραίους (int) και στη συνέχεια, με τη βοήθεια του new, δεσμεύουμε το χώρο που απαιτεί ένας ακέραιος (4 bytes). Στη δεύτερη γραμμή δηλώνουμε και πάλι έναν δείκτη προς ακεραίους. Αυτή τη φορά όμως δεσμεύουμε χώρο για δέκα ακεραίους (40 bytes). Εδώ αξίζει να σημειώσουμε το εξής: Ο χώρος που δεσμεύει *μία* εμφάνιση του τελεστή new είναι συνεχόμενος. Αυτό δεν ισχύει για τις ξεχωριστές εμφανίσεις του τελεστή. Στο παράδειγμα που είδαμε προηγουμένως, οι δέκα ακέραιοι θα βρίσκονται σε διαδοχικές θέσεις. Ωστόσο, ο χώρος για τον μεμονωμένο ακέραιο δεν είναι υποχρεωτικό ότι θα συνορεύει με την περιοχή της δεκάδας. Ας αυτό σας μπέρδεψε, σκεφτείτε την εξής (απλούστερη) περίπτωση:

```
int *ptr1 = new int;
int *ptr2 = new int;
```

Οι διευθύνσεις μνήμης στις οποίες παραπέμπουν οι δύο δείκτες μπορεί να είναι γειτονικές, αλλά μπορεί και να μην είναι.

Η δραστηριότητα μέσα στη στοίβα είναι αρκετά πιο έντονη απ' αυτή που υποψιαζόμαστε. Ας δούμε όσα συμβαίνουν στο σύστημα όταν καλείται μια συνάρτηση:

- Κατασκευάζεται ένα stack frame. Μπορείτε να το φαντάζεστε σαν μια νέα στοίβα, μέσα στη "μεγάλη" στοίβα του προγράμματος. Αρχικά, το stack frame περιλαμβάνει τα εξής:
  - Τη διεύθυνση μνήμης της εντολής, που ακολουθεί την κλήση της συνάρτησης. Ονομάζεται και return address, αφού εκεί θα επιστρέψει η ροή του προγράμματος όταν ολοκληρωθεί η συνάρτηση.
  - Τις τιμές ορισμένων καταχωριστών που ενδέχεται να τροποποιηθούν κατά την εκτέλεση της συνάρτησης και οι οποίοι αργότερα θα πρέπει να επανέλθουν στην αρχική τους κατάσταση.
  - Τις παραμέτρους που μεταφέρονται στη συνάρτηση.
- Η ροή του προγράμματος μεταφέρεται στο σημείο εκκίνησης της συνάρτησης και η εκτέλεσή της ξεκινά.
- Κάθε μεταβλητή που δηλώνεται και χρησιμοποιείται χωρίς να έχει δεσμευτεί χώρος χειροκίνητα (από τον προγραμματιστή), τοποθετείται στο stack frame της συνάρτησης.

- Όταν η συνάρτηση ολοκληρωθεί πραγματοποιούντα τα εξής:
  - Οι τοπικές μεταβλητές (όσες βρίσκονται στο stack frame) καθώς και οι παράμετροι που είχαν μεταφερθεί στη συνάρτηση, καταστρέφονται.
  - Οι καταχωριστές που ενδέχεται να είχαν αποθηκευτεί, ανακτούν τις αρχικές τους τιμές.
  - Η ροή του προγράμματος μεταφέρεται εκεί που υποδεικνύει το return address.

Οι τιμές που επιστρέφει μια συνάρτηση μεταφέρονται με διάφορους τρόπους. Σε ορισμένες περιπτώσεις αποθηκεύονται μέσα στο stack frame, ενώ σε κάποιες άλλες τοποθετούνται σε καταχωριστές του επεξεργαστή. Φυσικά, αυτές οι λεπτομέρειες δεν είναι κρίσιμες. Αρκεί να έχουμε υπόψη το γενικό τρόπο λειτουργίας της στοίβας και να θυμόμαστε ότι εκεί δεν τοποθετούνται μόνο μεταβλητές, αλλά και διευθύνσεις μνήμης που παραπέμπουν σε σημεία του κώδικα (return address).

Υπερχείλιση στοίβας

Όπως αναφέραμε νωρίτερα η στοίβα είναι φραγμένη – δεν μπορεί να εκτείνεται απεριόριστα. Για παράδειγμα, στα Windows το μέγεθος της στοίβας δεν μπορεί να ξεπερνά το 1MB, ενώ σε κάποια συστήματα Unix μπορεί να φτάνει έως τα 8MB. Όταν ένα πρόγραμμα προσπαθεί να τοποθετήσει στη στοίβα περισσότερα δεδομένα, συμβαίνει αυτό που ονομάζουμε υπερχείλιση στοίβας. Στην πράξη, τα πλεονάζοντα δεδομένα καταλήγουν σε περιοχές της μνήμης που προορίζονταν για άλλα, άσχετα δεδομένα. Κάτι τέτοιο, τουλάχιστον όταν συμβαίνει κατά λάθος, έχει απρόβλεπτα αποτελέσματα στη συμπεριφορά ενός προγράμματος και συνήθως οδηγεί στο "κρέμασμα". Υπερχείλιση στοίβας μπορούμε να προκαλέσουμε τεχνητά, με διάφορους τρόπους.

```
int main()
{
    int my_array[100000000];
    return 0;
}
```

Στο παραπάνω παράδειγμα επιχειρούμε να δεσμεύσουμε χώρο για *πάρα πολλές* μεταβλητές. Είναι πασιφανές ότι η στοίβα δεν μπορεί να χωρέσει όλα αυτά τα δεδομένα και το πρόγραμμα θα σκάσει αμέσως μόλις προσπαθήσουμε να το τρέξουμε. Δείτε κι αυτή την περίπτωση:

```
void foo()
{
    foo();
}
```

```
int main()
{
    foo();
    return 0;
}
```

Εδώ η συνάρτηση foo() καλείται αναδρομικά (δηλαδή από τον εαυτό της) χωρίς κάποιον περιορισμό. Ως αποτέλεσμα, θα αρχίσουν να δημιουργούνται αλλεπάλληλα stack frames, το ένα μέσα στο άλλο. Με αυτόν τον τρόπο ο διαθέσιμος χώρος για τη στοίβα θα εξαντληθεί πολύ γρήγορα. Γενικότερα, το παραπάνω πρόγραμμα θα καταναλώσει όλη τη διαθέσιμη μνήμη, μέχρι το πρόγραμμα να τερματιστεί.

Κρίσιμες λεπτομέρειες

Ας επιστρέψουμε στο θέμα μας τώρα. Η glibc, μέσα στη συνάρτηση _nss_dns_gethostbyname4_r() δεσμεύει 2048 bytes για την απάντηση στο εκάστοτε DNS query. Αυτός ο χώρος βρίσκεται μέσα στη στοίβα. Αντιθέτως, στις συναρτήσεις send_dg() και send_vc(), ο χώρος για την απάντηση στο εκάστοτε ερώτημα DNS δεσμεύεται δυναμικά, από το σωρό. Αυτό σημαίνει ότι δεσμεύονται ακριβώς τόσα bytes, όσα χρειάζονται για να χωρέσει η απάντηση. Αν το μέγεθός της υπερβαίνει τα 2048 bytes, θα δεσμευτεί περισσότερος χώρος *χωρίς* να προκύψει κάποιο πρόβλημα. Όλα αυτά μοιάζουν αδιάφορα, μόνο που οι παραπάνω συναρτήσεις χρησιμοποιούνται συνδυασμένα και μάλιστα σε πάρα πολλά προγράμματα. Υπό συγκεκριμένες συνθήκες, λοιπόν, μπορεί να συμβεί το εξής: Ο χώρος που έχει δεσμευτεί στο σωρό να είναι μεγαλύτερος από τον χώρο που έχει δεσμευτεί στη στοίβα. Αντιλαμβάνεστε τι σημαίνει αυτό; Μόλις μεταφερθούν τα δεδομένα από το σωρό στη στοίβα, θα πραγματοποιηθεί μια υπερχείλιση! Τα δεδομένα μετά το 2048 byte θα τοποθετηθούν σε άσχετες περιοχές της στοίβας – ή και εκτός αυτής.

Σενάρια επίθεσης

Η ευπάθεια που εξετάζουμε, θεωρητικά τουλάχιστον, επιτρέπει ακόμα και την εκτέλεση απομακρυσμένου κώδικα. Βέβαια κάνουμε λόγο για έναν θεωρητικό κίνδυνο, αφού στην πράξη είναι εξαιρετικά δύσκολο να συμβεί κάτι τέτοιο. Οι μηχανικοί της Google έχουν κατασκευάσει δύο προγράμματα που επιδεικνύουν την επίθεση μέσω αυτής της ευπάθειας και αποτελούν αυτό που ονομάζουμε "proof of concept". Πρόκειται για δύο προγράμματα, από τα οποία το ένα λειτουργεί σαν client και το άλλο σαν ένας υποτυπώδης name server. Το δεύτερο εκμεταλλεύεται την ευπάθεια, ενώ το πρώτο αποτελεί το θύμα. Με τη βοήθειά τους μπορούμε να τσεκάρουμε αν το σύστημά μας αντιμετωπίζει το –θεωρητικό τουλάχιστον– κίνδυνο. Μπορείτε να τα κατεβάσετε από το https://github.com/fjserna/CVE-2015-7547.

Skill: Intermediate
Tags: Unity, games, development

Η κινούμενη σφαίρα

Στο προηγούμενο τεύχος μιλήσαμε για τα indie games και κάναμε μια εισαγωγή στον κόσμο της Unity. Από αυτό το τεύχος ξεκινάμε με τη δημιουργία ενός παιχνιδιού – του δικού μας παιχνιδιού! Υποσχόμαστε ότι οι παρουσιάσεις μας θα αφορούν σε πολύ πιο πρακτικά θέματα, αραιά και πού ωστόσο θα κάνουμε μικρές παρενθέσεις εξηγώντας καλύτερα έννοιες που δεν θα έχουν αναλυθεί επαρκώς.

του Γιάννη Κρομμύδα

Δεν έχουμε βρει ακόμα κάποιο όνομα για το παιχνίδι μας και θα θέλαμε τις δικές σας προτάσεις. Αφήστε ένα σχόλιο στο site του περιοδικού με τ' όνομα που προτείνετε και ίσως γίνετε ο Godfather (TM) του παιχνιδιού! Κατά τα άλλα, έχουμε ήδη κατεβάσει την τελευταία έκδοση της μηχανής Unity (v5.3.4f1, τη στιγμή που γράφεται το παρόν) από τη διεύθυνση http://unity3d.com/get-unity/download?ref=personal και είμαστε έτοιμοι να ξεκινήσουμε. Εσείς;

Game Design Document τριών παραγράφων

Πριν περάσουμε στη Unity ας δούμε εν συντομία τι θα φτιάξουμε. Σκοπός του παίκτη σ' αυτό το πρώτο μας project, λοιπόν, θα είναι να περάσει από το ένα μέρος της πίστας στο άλλο *χωρίς* να χτυπήσει στα εμπόδια που βρίσκονται στην πορεία του.



Εικόνα 1.
Ένα γρήγορο προσχέδιο του παιχνιδιού, για να πάρετε μια γεύση του πώς πάνω κάτω θα φαίνεται το τελικό προϊόν. Δεν γίνεται να βαδίζουμε στα τυφλά. Γίνεται;

Ας δώσουμε λίγη ζωή στην ακίνητη εικόνα, εξηγώντας περαιτέρω. Ο παίκτης (κίτρινη σφαίρα) θα πρέπει να περάσει ανάμεσα από τα κινούμενα μαύρα εμπόδια και να φτάσει στην άλλη άκρη της πίστας – συγκεκριμένα στην ασπρόμαυρη γραμμή του τερματισμού. Η κίνηση του παίκτη θα γίνεται με τα βελάκια του δρομέα ή με τα πλήκτρα [W], [A], [S] και [D]. Η κίνηση αριστερά–δεξιά θα είναι ελεύθερη, αλλά για την κίνηση μπρος–πίσω θα γίνεται ένα μικρό άλμα που θα παίρνει συγκεκριμένο χρονικό διάστημα. Τα εμπόδια θα εμφανίζονται τυχαία, ανάλογα σε ποια γραμμή βρίσκονται. Θα κινούνται αριστερά ή δεξιά με διαφορετικές ταχύτητες, αλλά θα έχουν ίδια ταχύτητα και πορεία ανά γραμμή. Στο ενδιάμεσο, και για να κάνουμε τη ζωή του παίκτη λίγο πιο εύκολη, θα έχουμε νεκρές ζώνες (οι άσπρες οριζόντιες λωρίδες), που δεν θα έχουν εμπόδια. Έτσι, ο παίκτης θα χαλαρώνει και θα προετοιμάζεται για το επόμενο γκρουπ εμποδίων.

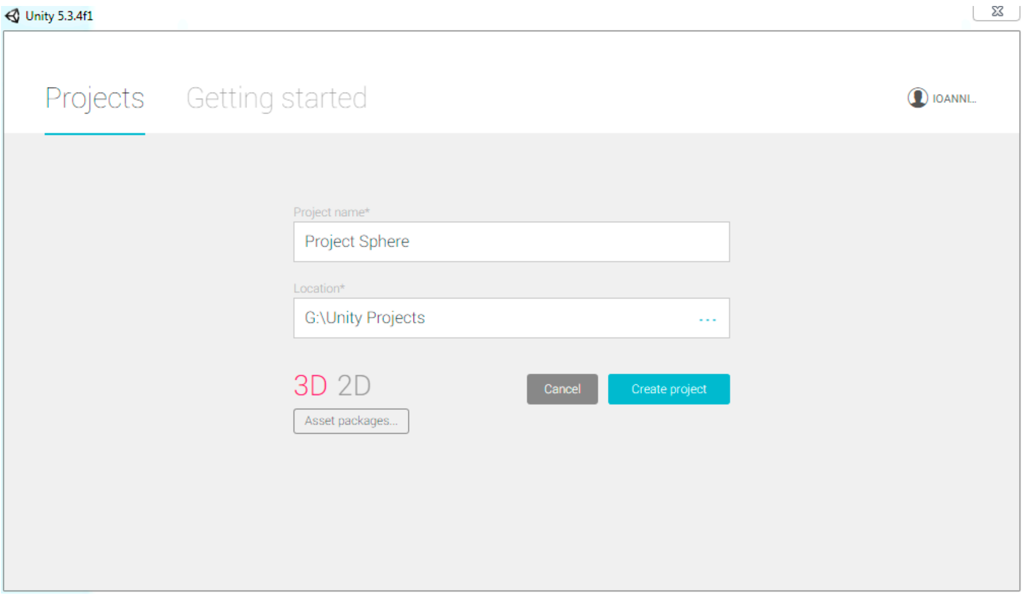
Αν ο παίκτης ακουμπήσει σε ένα εμπόδιο χάνει και θα πρέπει να παίξει την πίστα από την αρχή. Για να αποφύγουμε τις σπασμένες οθόνες (μια και ragequit δεν υπάρχει), θα του δίνουμε τη δυνατότητα, αν το θελήσει, να παίξει την εύκολη εκδοχή του παιχνιδιού. Σ' αυτή, όταν χάνει αντί να εμφανίζεται στην αρχή της πίστας θα εμφανίζεται στην προηγούμενη νεκρή ζώνη, με βάση την τρέχουσα θέση του.

Τώρα, επειδή παιχνίδι χωρίς κίνητρο δεν υφίσταται, θα μετράται ο χρόνος που έκανε ο παίκτης για να φτάσει από το ένα σημείο στο άλλο. Η εφαρμογή θα κρατάει τα τελικά αποτελέσματα σε ένα τοπικό (ίσως κι online) leaderboard, και θα τα εμφανίζει στο τέλος κάθε πίστας.

Ονομάζοντας τα projects

Ξεκινάμε τρέχοντας τη μηχανή Unity και δημιουργώντας ένα καινούργιο 3D project. Αποφασίζουμε πού θα το αποθηκεύσουμε και του δίνουμε ένα όνομα. Μα δεν έχουμε όνομα ακόμα, θα πείτε. Και θα σας πούμε ότι έχετε δίκαιο. Από τη στιγμή που δεν έχουμε όνομα και η Unity δεν μπορεί να κάνει χωρίς ένα όνομα για το project μας, μπορούμε να βάλουμε ό,τι θέλουμε. Αλλά, είναι σωστό αυτό; Ας δούμε τι γίνεται σε περίπτωση που δεν έχουμε όνομα για το project. Αν λοιπόν μας έχει πιάσει ο οίστρος γι' αυτήν την ωραία ιδέα gameplay που σκεφτήκαμε δευτερόλεπτα πριν, θα ήταν κακό να ξοδέψουμε χρόνο σκεπτόμενοι κάποιο όνομα για το μελλοντικό παιχνίδι που θα βασίζεται στην ιδέα. Πρέπει να μπούμε στο πετσί της διαδικασίας, δηλαδή να ξεκινήσουμε με την υλοποίηση των λειτουργιών του gameplay, άμεσα!

Στη βιομηχανία των παιχνιδιών και όχι μόνο, υπάρχουν τα λεγόμενα working titles. Ως working title χαρακτηρίζεται ο τίτλος ενός παιχνιδιού που χρησιμοποιείται κατά τη δημιουργία του – όχι κατ' ανάγκη και μετά. Υπάρχουν δύο λόγοι για την ύπαρξη των εν λόγω τίτλων. Ο πρώτος λόγος είναι, απλά, ότι δεν έχει ακόμα οριστικοποιηθεί ο τελικός τίτλος του εν λόγω παιχνιδιού. Σε αυτή την περίπτωση, του αποδίδεται ένα όνομα με μόνο σκοπό να γίνεται αναφορά σ' αυτό. Ο δεύτερος λόγος είναι λίγο πιο συνωμοτικός. Ενδέχεται κάποιο παιχνίδι να έχει ήδη τίτλο, αλλά η εταιρεία να θέλει να κρύψει την αληθινή ταυτότητά του από το ευρύ κοινό. Το εμφανίζει έτσι με το working title του, προσπαθώντας ταυτόχρονα να δημιουργήσει hype. Ας ονομάσουμε το έργο μας Project Sphere, μια και τον παίκτη παριστάνει μια σφαίρα. (Σαφώς μπορείτε να ονομάσετε όπως θέλετε το project στον υπολογιστή σας. Απλά, στην παρούσα σειρά άρθρων θα χρησιμοποιούμε αυτό το όνομα.)



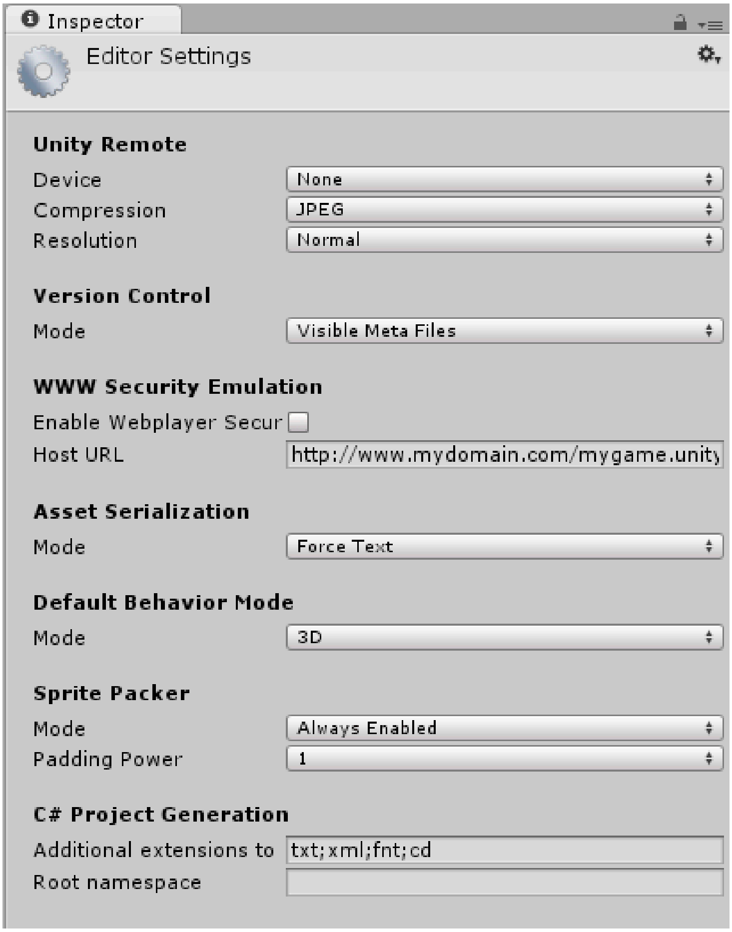
Εικόνα 2.
Μπορεί να θυμάστε την παραπάνω εικόνα δημιουργίας νέου project, από το προηγούμενο τεύχος. Βέβαια μπορεί και όχι. Αν ισχύει το δεύτερο, ίσως δεν έχετε διαβάσει τα δύο σχετικά άρθρα στο εν λόγω τεύχος. Σας πιάσαμε!

Source control, all the way

Προτού ξεκινήσουμε χρειάζεται να κάνουμε κάποιες ρυθμίσεις στη Unity, ώστε να μην υπάρχουν προβλήματα. Όπως αναφέραμε και σε προηγούμενο άρθρο της σειράς, θα χρησιμοποιήσουμε το Git και κατ' επέκτασιν το GitHub για το hosting του έργου μας. Η Unity, για κάθε asset που υπάρχει σε κάθε project της, χρειάζεται να αποθηκεύει και κάποια έξτρα δεδομένα για δική της χρήση. Τα δεδομένα αυτά αποθηκεύονται σε αρχεία τύπου meta. Όταν στη δημιουργία του παιχνιδιού εμπλέκονται περισσότερα από ένα άτομα, τα αρχεία αυτά πρέπει να έχουν ίδια δεδομένα σε κάθε μηχάνημα. Μια που εξ αρχής τα meta files των assets δεν είναι προσβάσιμα από τους χρήστες, πρέπει να πούμε στη Unity να μας τα δείξει.

Αυτό γίνεται από το μενού Edit → Project Settings → Editor. Στον Inspector της Unity θα δούμε μια σειρά από επιλογές όπως φαίνονται στο σχετικό screenshot.

Στο παράθυρο αυτό, στο Version Control Mode επιλέγουμε Visible Meta Files. Από τη στιγμή που δεν έχουμε ακόμα assets στο project μας, η αλλαγή είναι στιγμιαία. Αν είχαμε, ανάλογα με το μέγεθος του project θα χρειαζόταν να περιμένουμε λίγη ώρα, ώστε η Unity να δημιουργήσει meta files για όλα τα assets που υπάρχουν στο project. Μέσα σ' ένα project, meta files έχουν ακόμα και οι φάκελοι.



Εικόνα 3.
Όλα τα settings που μπορούμε να αλλάξουμε στη Unity εμφανίζονται σε κατηγορίες κι εμφανίζονται στο παράθυρο του Inspector.

Αφού βρισκόμαστε στις ρυθμίσεις, ας δούμε και το asset serialization mode. Με την εν λόγω επιλογή μπορούμε να πούμε στη Unity αν θα αποθηκεύει τις σκηνές μας σε text ή σε binary mode. Το binary mode καταλαμβάνει λιγότερο χώρο στο δίσκο, αλλά καθιστά αδύνατο το merge ενός αρχείου, αν αυτό έχει τροποποιηθεί από περισσότερα του ενός άτομα. Το text mode αντίθετα αποθηκεύει τα δεδομένα της σκηνής σε YAML. Αν και έτσι καταλαμβάνουν αρκετά περισσότερο χώρο στο δίσκο, καθίσταται δυνατή η εργασία σε μία σκηνή από πολλά άτομα, παράλληλα (το merge, αργότερα, είναι εφικτό). Η τρίτη επιλογή επιτρέπει και τους δύο τύπους αρχείων σκηνών για το ίδιο project. Στις άλλες δύο περιπτώσεις, αν η Unity αντιληφθεί ένα αρχείο σκηνής σε διαφορετικό mode, θα το μετατρέψει αυτόματα στο επιλεγμένο.

Let the games begin!

Τακτοποιήσαμε λοιπόν λίγες ακόμα από τις ιδιοτροπίες της Unity, οπότε είναι καιρός ν' αρχίσουμε να δημιουργούμε το παιχνίδι μας. Αρχικά θα φτιάξουμε τη σκηνή που θα λαμβάνει χώρα το κύριο μέρος του παιχνιδιού (εκεί που θα κινείται ο παί-

κτης). Έχει αρκετά πράγματα προς υλοποίηση, συνεπώς αξίζει να τη χωρίσουμε σε μικρότερα κομμάτια. Ας αρχίζουμε από τον παίκτη – ή ακριβέστερα από τη σφαίρα που θα κινεί ο παίκτης.

Στο παράθυρο του νέου μας project η Unity παρουσιάζει μια κενή σκηνή. Για την ακρίβεια, δεν είναι 100% κενή: περιέχει μια κάμερα (Main Camera) αλλά και μια κατευθυντήρια πηγή φωτός (Directional Light). Πρόκειται βέβαια για την προκαθορισμένη σκηνή της Unity, οπότε κι εμείς καταχρηστικά την αναφέρουμε ως "κενή".

Στο παράθυρο του Project και με δεξί κλικ στον φάκελο Assets, επιλέγουμε Create → Folder και δημιουργούμε τέσσερις φακέλους με τα εξής ονόματα: Materials, Scenes, Scripts και Resources. Πιστεύουμε ότι τα ονόματα των φακέλων προϊδεάζουν για το τι μπορεί να περιέχει ο καθένας – εκτός ίσως από το όνομα του τελευταίου. Σε κάθε περίπτωση και για να είμαστε εντάξει, οφείλουμε να είμαστε αναλυτικοί.

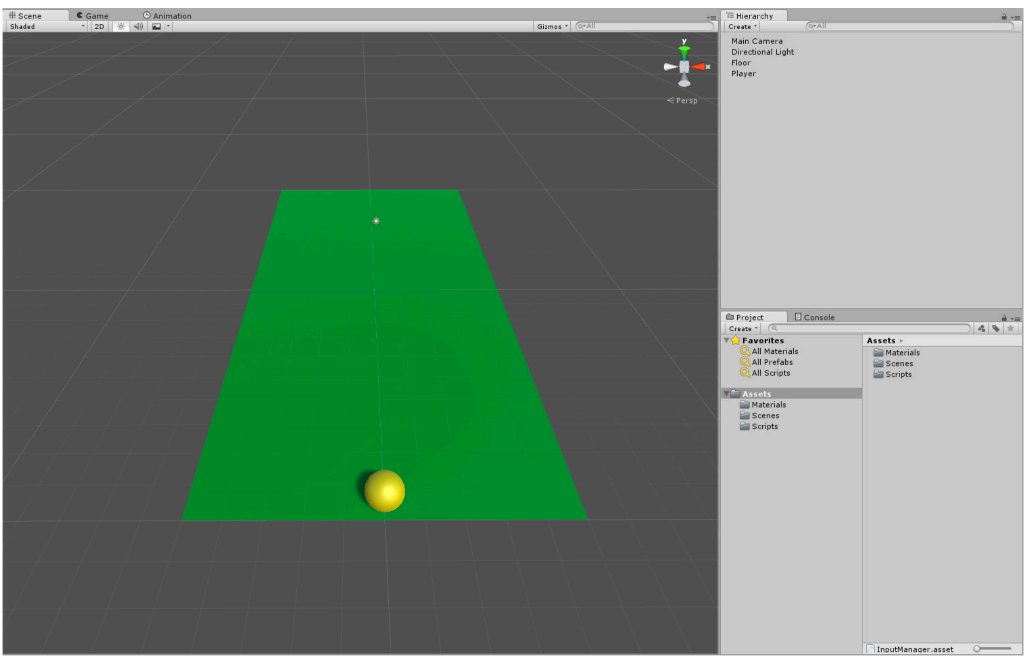
Ο πρώτος φάκελος, λοιπόν, θα περιέχει όλα τα materials που θα χρησιμοποιούμε για να δώσουμε χρώμα στα αντικείμενα της σκηνής μας. Ο δεύτερος θα περιέχει όλες τις σκηνές που θα χρειαστεί το παιχνίδι μας, ενώ ο τρίτος τα C# scripts για τις λειτουργίες του παιχνιδιού. Ο τελευταίος φάκελος έχει ιδιαίτερη σημασία σε ένα project. Σε αυτόν μπορούμε να αποθηκεύουμε GameObjects που δεν βρίσκονται ήδη στη σκηνή και να δημιουργούμε instances τους μέσω scripts, κατά τη διάρκεια εκτέλεσης του παιχνιδιού.

Πίσω στη δημιουργία της πίστας όμως.

- Πηγαίνουμε στο μενού GameObject → 3D Object → Plane ώστε να δημιουργήσουμε ένα plane που θα χρησιμοποιηθεί ως το δάπεδο της πίστας.
- Στο παράθυρο Hierarchy το μετονομάζουμε σε "Floor" (με το [F2] αφού το επιλέξουμε ή με δεξί κλικ/Rename ή από τον Inspector – πάντα αφού το επιλέξουμε).
- Στον Inspector αυξάνουμε το Scale του άξονα z στο 2 (Scale = (1, 1, 2)).
- Αν δεν είναι ήδη έτσι, ως Position και Rotation θέτουμε μηδενικές τιμές. Αυτό μπορεί να γίνει είτε αλλάζοντας τις τιμές από τα πεδία του Inspector, είτε κάνοντας κλικ στο γρανάζι (βλ. πάνω δεξιά γωνία του Transform Component) κι επιλέγοντας Reset Position ή Reset Rotation αντίστοιχα.
- Στον φάκελο Materials στο Project, δημιουργούμε ένα Material (δεξί κλικ στον φάκελο και Create → Material) και τ' ονομάζουμε "Floor".
- Με επιλεγμένο το material στον Inspector, κάνουμε κλικ στο άσπρο κουτάκι δεξιά του Albedo και του δίνουμε ένα σκούρο πράσινο χρώμα (#0A5D00FF).
- Σέρνουμε το material που έχουμε μόλις δημιουργήσει από το παράθυρο του Project, είτε στο αντικείμενο που βλέπουμε στη σκηνή είτε στο όνομα του αντικειμένου στο παράθυρο Hierarchy. Αν σύρουμε το material στη σκηνή, θα δούμε ότι όταν το ποντίκι βρίσκεται πάνω από ένα αντικείμενο που μπορεί να το δεχτεί, η αλλαγή γίνεται επί τόπου. Μπορούμε να την επιβεβαιώσουμε αφήνοντας το ποντίκι μας.

Πλέον έχουμε ένα πράσινο έδαφος. Ας συνεχίσουμε με την σφαίρα που θα δείχνει τον παίκτη μας.

- Δημιουργούμε μια σφαίρα από το μενού GameObject → 3D Object → Sphere, με όνομα "Player".
- Στον Inspector θέτουμε Position = (0, 0.5, -9.5) και Rotation = (0, 0, 0).
- Με τη διαδικασία που αναφέραμε ήδη, και στον φάκελο Materials, δημιουργούμε ένα νέο material με όνομα "Player" καθώς κι ένα σκούρο κίτρινο χρώμα (#A69400FF).
- Το σέρνουμε στη σφαίρα του παίκτη ώστε να το ορίσουμε ως material της. Αποθηκεύουμε τη σκηνή μας στον φάκελο Scenes και με όνομα Game.unity. Πάμε ολοταχώς να γράψουμε το πρώτο μας script.
- Με δεξί κλικ στον φάκελο Scripts, επιλέγουμε Create → C# Script και του δίνουμε τ' όνομα "Player".



Εικόνα 4.
Η σκηνή μας, έτοιμη προς αποθήκευση!

Αρχικά, το Script αυτό θα είναι υπεύθυνο για την κίνηση του παίκτη. Με διπλό κλικ ανοίγουμε τον editor που έχουμε επιλέξει και προσθέτουμε τον παρακάτω κώδικα.

```
using UnityEngine;
namespace Deltahacker
```

```
{
      public class Player : MonoBehaviour
      {
         public float Speed = 3f;
         public void Update()
         {
            float horizontal = Input.GetAxis("Horizontal");
            Vector3 movement = new Vector3(horizontal * Speed * Time.deltaTime,
            0, 0);
            transform.position += movement;
         }
      }
}
```

Στο σημείο αυτό επιτρέψτε μας μερικές διευκρινίσεις. Αρχικά, το namespace definition δεν είναι αναγκαίο. Όσοι δεν είστε εξοικειωμένοι με τη χρήση namespaces στον κώδικά σας, π.χ., διότι η γλώσσα που χρησιμοποιείτε δεν τα υποστηρίζει, μπορείτε να το παραλείψετε. Γενικά είναι πολύ καλή τακτική, για διάφορους λόγους. Η αιτιολόγηση ξεφεύγει από το σκοπό του παρόντος άρθρου, οπότε σταματάμε εδώ λέγοντας μόνο ότι θα βλέπετε namespaces σχεδόν πάντα στον κώδικά μας.

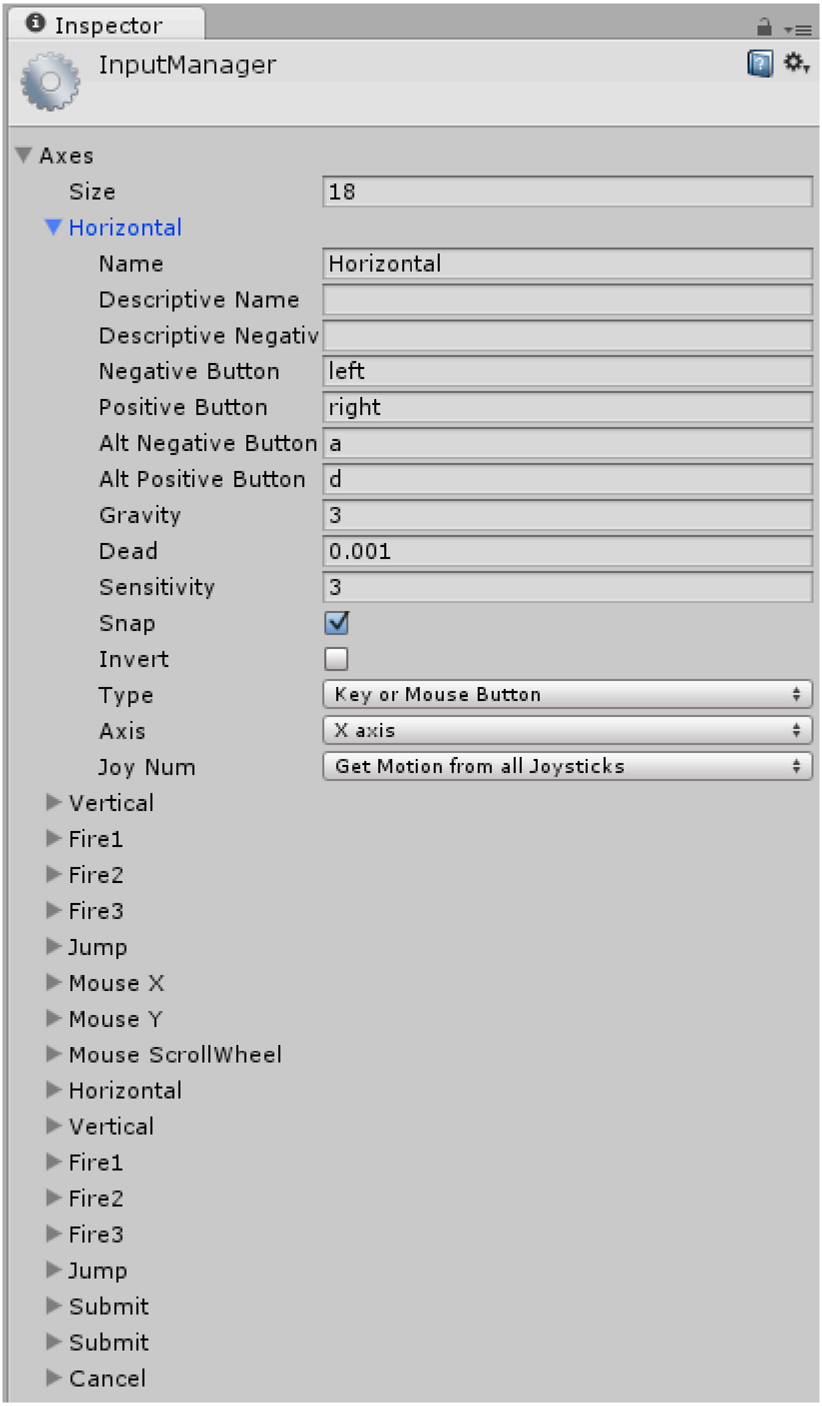
Η μέθοδος Update της κλάσης είναι η προκαθορισμένη της Unity και καλείται σε συγκεκριμένα σημεία από την ίδια τη μηχανή. Πιο συγκεκριμένα, εκτελείται σε κάθε frame λειτουργίας του παιχνιδιού. Θα μιλήσουμε εκτενέστερα για τα frames σε επόμενο άρθρο. Το Game Loop pattern είναι σήμα κατατεθέν των παιχνιδιών και δεν γίνεται να μην κάνουμε μια αναφορά σ' αυτό.

Μέσα στη μέθοδο, στη μεταβλητή horizontal αποθηκεύουμε το Input του παίκτη. Βασιζόμενοι στο Input δημιουργούμε ένα νέο τρισδιάστατο διάνυσμα που ορίζει τη μετατόπιση της σφαίρας, την ορισμένη ταχύτητα και την ώρα που έχει περάσει από το προηγούμενο frame, ώστε να έχουμε σταθερή ταχύτητα χωρίς να μας ενδιαφέρει αν το παιχνίδι τρέχει με πολλά ή με λίγα frames.

User Input, κανείς;

Στο script αυτό επιλέγουμε να διαβάσουμε την είσοδο από τον παίκτη, μέσω του άξονα με όνομα Horizontal. Κάθε νέο project έχει προκαθορισμένους ένα σύνολο από άξονες, με τους οποίους μπορούμε να διαβάζουμε την είσοδο του παίκτη.

Τους εν λόγω άξονες μπορούμε να βλέπουμε από το μενού Edit → Settings → Input και, με expand στον Inspector, βλέπουμε και τη λίστα Axes. Αν κάνουμε κλικ στο entry ονόματι Horizontal, θα δούμε πώς ορίζεται αυτός ο άξονας.



Εικόνα 5.
Οι προκαθορισμένοι άξονες εισόδου για το παιχνίδι μας, όπως τους έχει ορίσει η Unity.

Τα κυριότερα σημεία είναι τα Negative και Positive Buttons, καθώς και τα εναλλακτικά τους (Alt Negative Button και Alt Positive Button). Στη συγκεκριμένη περίπτωση βλέπουμε ότι ο άξονας αυτός παίρνει θετικές τιμές αν πατήσουμε το δεξί βελάκι του πληκτρολογίου ή το πλήκτρο [D], και αρνητικές τιμές με το αριστερό βελάκι ή με το πλήκτρο [A] αντίστοιχα. Αν παρατηρήσουμε τη λίστα, θα δούμε ότι υπάρχει ακόμα μια εγγραφή με το ίδιο όνομα. Η δεύτερη εγγραφή είναι για χρήστες που χρησιμοποιούν joystick. Το αποτέλεσμα του άξονα, σε περίπτωση που χρησιμοποιούμε τα κουμπιά, είναι οι διακριτές τιμές -1, 0 και 1. Σε περίπτωση που χρησιμοποιήσουμε joystick οι τιμές έχουν εύρος [-1, 1], ανάλογα με την κλίση ή τη θέση του χειριστηρίου. Ας επιστρέψουμε όμως στο project μας.

Move baby, move!

Μια και το script είναι έτοιμο, πρέπει να το προσθέσουμε στη σφαίρα μας. Αυτό γίνεται με δύο τρόπους. Ο πρώτος είναι να σύρουμε την εγγραφή του από το παράθυρο του project στη σφαίρα, στο παράθυρο Hierarchy (ή στη σκηνή κατευθείαν). Εναλλακτικά, επιλέγουμε το αντικείμενο από το παράθυρο Hierarchy και πατάμε το κουμπί Add Component στον Inspector. Αν το κάνουμε με τον δεύτερο τρόπο, μπορούμε είτε να ψάξουμε το όνομα του script από το search box που εμφανίζεται είτε να το βρούμε δίνοντας Scripts → Deltahacker → Player.

Πριν δούμε τον κώδικά μας εν δράσει, έχουμε μια ακόμα μικρή λεπτομέρεια να διευθετήσουμε. Πρέπει συγκεκριμένα να τοποθετήσουμε την κάμερα σε ένα καλό σημείο, ώστε να φαίνεται όλη η πίστα κι ο παίκτης να βρίσκεται στο κάτω μέρος της.

- Γυρίστε το viewport της σκηνής σε μια καλή θέση (όπως, π.χ., φαίνεται στην εικόνα 3) κι επιλέξτε την κάμερα (Main Camera) από το Hierarchy.
- Για να κάνουμε την κάμερα να δείχνει το σημείο που βλέπουμε μέσω του viewport της σκηνής μας, πρέπει να πάμε στο μενού GameObject → Align With View ή να πατήσουμε το συνδυασμό πλήκτρων [CTRL+SHIFT+F].

Προσοχή στην επιλογή του αντικειμένου της κάμερας! Αν δεν το επιλέξετε μπορεί να αλλάξετε τη θέση και την περιστροφή κάποιου άλλου αντικειμένου. Αν γίνει αυτό, ένα [CTRL+Z] θα λύσει το πρόβλημα.

Αφού η κάμερα έχει μετακινηθεί, αποθηκεύουμε τη σκηνή μας και τρέχουμε το παιχνίδι πατώντας [CTRL+P] ή επιλέγοντας Edit → Play ή με κλικ στο κουμπί Play, στο κέντρο του πάνω μέρους του παραθύρου της Unity. Εν ώρα εκτέλεσης, αν πατήσουμε τα [A], [D] ή το δεξί/αριστερό βελάκι, θα δούμε τη σφαίρα να κινείται αριστερά-δεξιά.

Ως άσκηση, σας προτρέπουμε να μετακινήσετε τη σφαίρα στο κέντρο του πράσινου δαπέδου και να προσπαθήσετε να την κάνετε να κινείται και πάνω-κάτω. Ενδεικτικά, αναφέρουμε ότι πρέπει να πάρετε την είσοδο του παίκτη από τον κάθετο άξονα και να προσθέσετε το αποτέλεσμα στο διάνυσμα της κίνησης. Ο κώδικας της λύσης θα ανεβεί ως σχόλιο του άρθρου, στο site του περιοδικού. Τον κώδικα του project θα τον βρείτε στο GitHub και στη διεύθυνση https://github.com/ikromm/project-sphere.

Την επόμενη φορά θα μιλήσουμε για το Game Loop και στη συνέχεια θα κάνουμε τη σφαίρα να πηδάει μπρος και πίσω, ώστε να μπορεί ν' αλλάζει λωρίδες στο τελικό μας game.


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

Skill: Intermediate
Tags: jumplists, PRNG, LFSR, MML, tracker

Κατασκευάστε έναν υπολογιστή, μέρος 6

Μουσικές γλώσσες

Το synthesizer είναι έτοιμο, αλλά η κάρτα ήχου συμπεριφέρεται σαν εκείνα τα καρτ-ποστάλ, που όταν τα ανοίγεις παίζουν ξανά και ξανά την ίδια μελωδία. Απαιτείται οπωσδήποτε ένας μηχανισμός για την επικοινωνία της κάρτας με τον υπολογιστή. Πολύ περισσότερο, χρειαζόμαστε ένα εργαλείο που θα επιτρέπει στους μουσικούς να συνθέτουν αριστουργήματα, ειδικά για την κάρτα μας!

του Παναγιώτη Βαρελά

Όπως συμβαίνει με την ανάπτυξη κάθε προγράμματος, το firmware του synthesizer δεν γράφτηκε μονομιάς. Χρειάστηκαν αμέτρητες δοκιμές, κατά τη διάρκεια των οποίων συγκεντρώναμε όλη μας την προσοχή στους παράξενους ήχους που έβγαζαν τα ηχεία. Κάθε φορά ζητούσαμε την αναπαραγωγή της κατάλληλης αλληλουχίας από νότες και μ' αυτόν τον τρόπο τσεκάραμε τις διάρκειες και τους ρυθμούς, τις οκτάβες, τα εφέ, την ταυτόχρονη αναπαραγωγή σε πολλά κανάλια και πάει λέγοντας. Οι δοκιμαστικές μελωδίες τοποθετούνταν σε μια περιοχή της μνήμης flash κι αποτελούσαν τμήμα του προγράμματος (ήταν hard-coded). Με άλλα λόγια, για να τροποποιήσουμε έστω και μία νότα, έπρεπε να εγκαταστήσουμε νέο firmware στον μικροελεγκτή. Αυτός ο τρόπος εργασίας καθυστερούσε σημαντικά τις δοκιμές, αλλά δεν υπήρχε καμία εναλλακτική. Σ' εκείνη τη φάση το κύκλωμα ήχου δεν μπορούσε να επικοινωνήσει με τον υπολογιστή. Κατά συνέπεια, δεν μπορούσε να λάβει πληροφορίες για κάποια νότα, ούτε και γενικότερες οδηγίες του στιλ "σταμάτα" ή "ξεκίνα" την αναπαραγωγή. Το ίδιο ζήτημα είχαμε αντιμετωπίσει και κατά την ανάπτυξη του κυκλώματος γραφικών. Τελειοποιώντας τον κώδικα που παρήγαγε το σήμα VGA, βρισκόμασταν σ' ένα στάδιο όπου –τουλάχιστον θεωρητικά– μπορούσαμε να προβάλουμε στην οθόνη οτιδήποτε. Ωστόσο, το κύκλωμά μας εμφάνιζε μόνο ένα συγκεκριμένο μήνυμα, καρφωμένο στο firmware. Η κάρτα γραφικών ολοκληρώθηκε όταν προσθέσαμε τις ρουτίνες για την επικοινωνία με τον υπολογιστή. Έτσι συνέβη και με την κάρτα ήχου.

Τελική συναρμολόγηση

Αν και το καθυστερήσαμε, πιστεύουμε ότι πρέπει να πάρετε τον κώδικα που γράψαμε για την κάρτα ήχου. Το σχετικό πακέτο βρίσκεται στο http://bit.ly/dh053sndfirm. Σημειώστε ότι ο κώδικας είναι γραμμένος για τον assembler της ATMEL, που τρέχει μόνο σε Windows. Οι φίλοι που έχουν αποκλειστικά Linux, μπορούν να χρησιμοποιήσουν τον συμβατό assembler ονόματι avra.

Το synthesizer που εξετάσαμε στο προηγούμενο άρθρο αποτελεί το σημαντικότερο εξάρτημα της κάρτας ήχου. Κι αν λάβουμε υπόψη ότι πρόκειται για πρόγραμμα που τρέχει σ' έναν μικροελεγκτή χρονισμένο στα 16MHz, οι δυνατότητες που προσφέρει δεν είναι λίγες. Θα περίμενε κανείς ότι αυτή η διαπίστωση θα εκτόξευε το ηθικό μας στα ύψη. Ε, λοιπόν, στην πράξη μάς προκάλεσε έναν ελαφρύ πονοκέφαλο. Βρισκόμασταν στη φάση που το synthesizer δούλευε άψογα και ο ήχος του ήταν πεντακάθαρος (γκουχ, γκουχ), μόνο που η κάρτα ήχου δεν είχε γίνει ακόμα ...κάρτα. Χρειαζόταν έναν κώδικα επικοινωνίας για να δέχεται εντολές από τον επεξεργαστή του υπολογιστή μας. Σε μια προσπάθεια να οργανώσουμε τη δουλειά και να περιορίσουμε τα μελλοντικά προβλήματα στην ανάπτυξη του κώδικα, ξεκινήσαμε καταγράφοντας τις ανάγκες μας.

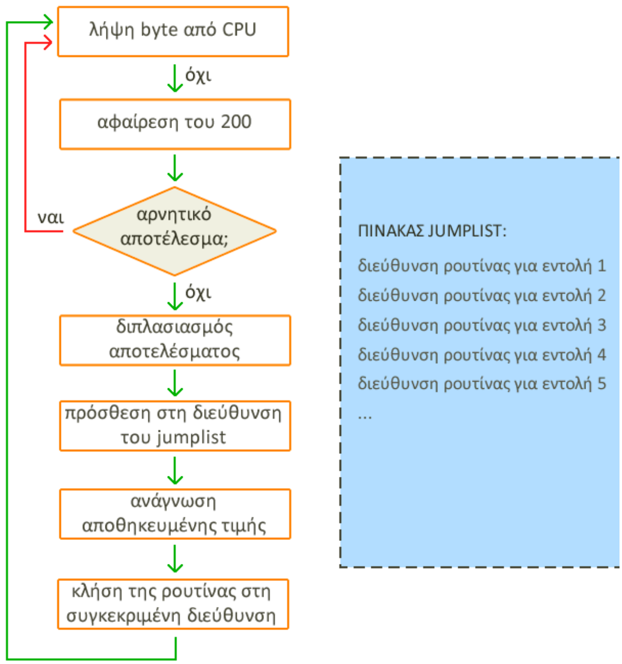
Η κάρτα ήχου δεν μπορεί να αποφασίζει μόνη της πότε θα παίζει μουσική και πότε θα σταματά. Αυτό είναι αυτονόητο, αλλά έπρεπε να καταγραφεί για να ενταχθεί στο γενικότερο σχεδιασμό. Σκεφτείτε τώρα το εξής: Τι πρέπει να κάνει ο επεξεργαστής, όταν επενδύουμε μουσικά ένα πρόγραμμα; Μήπως πρέπει να στέλνει τις νότες ξεχωριστά κι όταν ολοκληρώνεται η αναπαραγωγή της μίας, να στέλνει την επόμενη; Αν συνέβαινε κάτι τέτοιο, η εκτέλεση μιας μελωδίας θα απαιτούσε την πλήρη προσοχή του επεξεργαστή και το σύστημα δεν θα μπορούσε να κάνει τίποτα άλλο. Περιττό να πούμε ότι αυτή η συμπεριφορά θα ήταν απαράδεκτη ακόμα και για τον ταπεινό υπολογιστή μας. Γι' αυτό το λόγο καταστρώσαμε έναν μηχανισμό αποθήκευσης και διαχείρισης της μουσικής. Κατ' αρχάς, χωρίσαμε τη διαθέσιμη μνήμη της κάρτας ήχου (αναφερόμαστε στην SRAM του atmega328) σε τέσσερεις περιοχές. Εκεί αποφασίσαμε ότι θα αποθηκεύονται οι νότες που στέλνει ο επεξεργαστής, ανάλογα με το κανάλι για το οποίο προορίζονται. Έτσι, το synthesizer θα παίρνει τις διαδοχικές νότες από τη μνήμη και η εκάστοτε μελωδία θα ξετυλίγεται χωρίς τη διαρκή παρέμβαση του επεξεργαστή. Μήπως αυτό σημαίνει ότι πρέπει να ακούγονται πάντα όλα τα κανάλια; Κάποιες μελωδίες θα αξιοποιούν ένα κανάλι, ενώ άλλες θα αξιοποιούν περισσότερα. Τα περισσευούμενα κανάλια δεν πρέπει να ακούγονται, ανεξάρτητα από το αν υπάρχουν νότες στις αντίστοιχες περιοχές της μνήμης. Αυτές οι νότες ενδέχεται να έχουν ξεμείνει από κάποια προηγούμενη μελωδία ή να έχουν τοποθετηθεί για κάποια μελωδία που θα παιχτεί αργότερα.

Η παραπάνω παράγραφος ενδέχεται να προκάλεσε ελαφρύ πονοκέφαλο και σε σας. Το σύστημα που πρέπει να στήσουμε όμως δεν διαφέρει ιδιαίτερα από εκείνο της κάρτας γραφικών. Η μόνη ουσιαστική διαφορά είναι ότι σ' εκείνη την περίπτωση διαχειριζόμασταν έναν αποθηκευτικό χώρο (τη frame buffer), ενώ στην κάρτα ήχου θα έχουμε τέσσερεις τέτοιους χώρους. Κατά τα άλλα, η λογική των δύο προγραμμάτων δεν διαφέρει ιδιαίτερα. Ο κεντρικός βρόχος του προγράμματος βρίσκεται στο αρχείο snd-fw.asm και ξεκινά στη γραμμή 195. Το πρώτο πράγμα που συμβαίνει εκεί είναι η κλήση της get_byte. Η συγκεκριμένη συνάρτηση αναλαμβάνει το χειρισμό των ακροδεκτών, που εμπλέκονται στην επικοινωνία με τον επεξεργαστή. Η λειτουργία της τερματίζεται όταν ληφθεί κάποιο byte και η σχετική τιμή τοποθετείται στον καταχωριστή tmp1. Στη συνέχεια τσεκάρουμε αν η τιμή του tmp1 βρίσκεται εντός αποδεκτών ορίων. Οι εντολές που υποστηρίζονται είναι μόλις οκτώ και τις έχουμε αντιστοιχίσει στις τιμές από 200 έως 207. Αμέσως μετά χρησιμοποιούμε ένα jump list για να καλέσουμε την κατάλληλη συνάρτηση. Μπορείτε να διαβάσετε

περισσότερα για τα jump lists στο τρίτο μέρος της σειράς μας, στο τεύχος 048. Ας ρίξουμε μια ματιά στις συναρτήσεις που περιλαμβάνει το jump list και θα κατανοήσετε πλήρως τη λογική του προγράμματός μας.

- **xenable / xdisable.** Το όνομά τους δεν αφήνει περιθώρια παρεξήγησης. Αν εξετάσετε τον κώδικα των δύο συναρτήσεων, θα διαπιστώσετε ότι περιλαμβάνουν μια κλήση στη get_byte. Με αυτόν τον τρόπο λαμβάνουν ένα πρόσθετο byte από τον επεξεργαστή: την παράμετρο που δηλώνει ποιο κανάλι θα ενεργοποιηθεί ή απενεργοποιηθεί.
- **xclear.** Η συγκεκριμένη συνάρτηση διαγράφει την περιοχή μνήμης που χρησιμοποιείται για τις νότες ενός καναλιού. Η απαιτούμενη παράμετρος –το κανάλι που θα διαγραφεί– λαμβάνεται και πάλι με τη βοήθεια της get_byte.
- **xtempo.** Η εν λόγω συνάρτηση δεν κάνει τίποτα περισσότερο από το να λαμβάνει την τιμή του επιθυμητού ρυθμού και να την αποθηκεύει στον καταχωριστή rhythm. Εκεί ανατρέχει και ο μηχανισμός του synthesizer, για να πληροφορηθεί περί του επιλεγμένου tempo.
- **xnotes.** Εδώ τα πράγματα περιπλέκονται λίγο, αλλά μην περιμένετε τίποτα φοβερό. Αποστολή αυτής της συνάρτησης είναι η λήψη μιας νότας και η αποθήκευσή της στην κατάλληλη περιοχή μνήμης, σύμφωνα με το κανάλι για το οποίο προορίζεται. Εδώ η get_byte καλείται τρεις φορές: Μία για τη λήψη του καναλιού και άλλες δύο για τη λήψη της νότας (όλες οι νότες περιγράφονται πάντα από δύο bytes). Ανάμεσα σ' όλα αυτά θα εντοπίσετε και μια κλήση στη συνάρτηση find_marker, η οποία εντοπίζει την τελευταία νότα στην περιοχή μνήμης του επιλεγμένου καναλιού. Με τη βοήθειά της, μαθαίνουμε σε ποια διεύθυνση μνήμης πρέπει να αποθηκευτεί η νέα νότα.
- **xplay, xstop.** Η πρώτη ξεκινά το synthesizer (καλεί τη συνάρτηση play), ενώ η δεύτερη δεν κάνει απολύτως τίποτα. Αυτό μπορεί να μοιάζει παράξενο, αλλά αν το καλοσκεφτείτε είναι αναμενόμενο. Εφόσον βρισκόμαστε στον κύριο βρόχο του προγράμματος, το synthesizer δεν έχει ξεκινήσει και η κάρτα ήχου δεν αναπαράγει τίποτα. Η εντολή της διακοπής δεν έχει κανένα νόημα σε αυτή τη φάση. Αν παρ' όλα αυτά τη στείλει ο επεξεργαστής, το πρόγραμμα πρέπει να την αγνοήσει. Ο ουσιαστικός έλεγχος για τη λήψη αυτής της εντολής πραγματοποιείται μέσα στον κεντρικό βρόχο του synthesizer.

Ο κώδικας των παραπάνω συναρτήσεων βρίσκεται στο αρχείο api.asm. Για το τέλος αφήσαμε μια απορία δευτερεύουσας σημασίας: Αναρωτιέστε γιατί τα ονόματα των συναρτήσεων ξεκινούν με το γράμμα x; Γι' αυτό ευθύνεται εν μέρει η παραξενιά του γράφοντα, αλλά υπήρχε κι ένας τεχνικός λόγος. Στην Assembly δεν γίνεται διάκριση μεταξύ των ονομάτων των συναρτήσεων και των labels μέσα στον κώδικα. Για να μην προκαλούνται μπερδέματα, ο προγραμματιστής οφείλει να χρησιμοποιεί διαφορετικά ονόματα. Το γράμμα x προστέθηκε για να αποφύγουμε το conflict με ετικέτες του τύπου play, stop, tempo κ.ά, που ήταν πολύ πιθανό να έχουμε χρησιμοποιήσει στον κώδικα του synthesizer. Ένα τέτοιο λάθος δεν θα περνούσε απαρατήρητο (ο assembler θα έσκουζε!) αλλά θεωρήσαμε φρόνιμο ν' αποφύγουμε αυτά τα μπερδέματα μ' έναν "ομοιόμορφο" τρόπο.



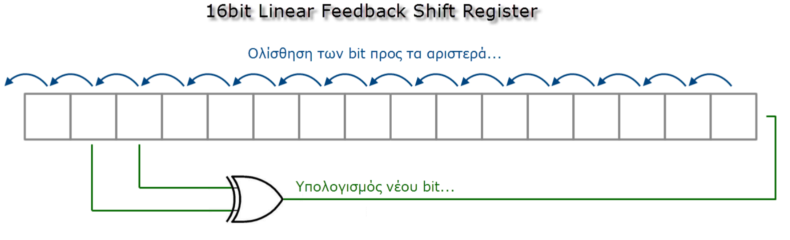
Ο κεντρικός βρόχος του firmware της κάρτας ήχου. Όπως και στην περίπτωση της κάρτας γραφικών, το πρόγραμμα περιμένει εντολές από τον επεξεργαστή. Όταν λαμβάνεται ένα byte πραγματοποιούνται ορισμένοι έλεγχοι, για να διαπιστωθεί αν πρόκειται για έγκυρη εντολή. Εφόσον όλα δείχνουν σωστά, αξιοποιείται ένα jumplist για την κλήση της κατάλληλης συνάρτησης. Χωρίς το jumplist, ο κώδικας θα περιλάμβανε μια (άκομψη) σειρά από διαδοχικά IF.

Προβλέψιμος θόρυβος

Στο προηγούμενο άρθρο της σειράς εξετάσαμε τη λειτουργία του synthesizer με κάθε λεπτομέρεια. Ωστόσο, πέρα από μια γρήγορη αναφορά στο κανάλι του θορύβου, δεν είπαμε τίποτα συγκεκριμένο για τη λειτουργία του. Όπως μπορεί να υποθέσει κανείς, τα δείγματα γι' αυτό το κανάλι αποτελούν τυχαίες αριθμητικές τιμές. Απλά πράγματα, θα πείτε, μόνο που το πρόγραμμά μας είναι γραμμένο σε Assembly. Δεν υπάρχει καμία έτοιμη συνάρτηση που να λύνει αυτό το πρόβλημα. Πώς υπολογίζονται οι τυχαίες τιμές; Βλέπετε την αντίφαση στην προηγούμενη ερώτηση; Στην πραγματικότητα, τα δείγματα για το κανάλι του θορύβου δεν είναι πραγματικά τυχαία. Πρόκειται για τιμές που μεταβάλλονται κατά τέτοιον τρόπο, ώστε η διαδοχή τους να προσλαμβάνεται από την αντίληψή μας ως απρόβλεπτη. Ακριβώς γι' αυτό, οι τιμές του είδους χαρακτηρίζονται ψευδοτυχαίες και οι μηχανισμοί που τις παράγουν ονομάζονται γεννήτριες ψευδοτυχαίων αριθμών (PRNG, από το Pseudo-Random Number Generator). Για την υλοποίηση ενός PRNG προσφέρονται πολλοί αλγόριθμοι. Εμείς επιλέξαμε κάποιον που απαιτεί ελάχιστους και απλούς υπολογισμούς, γεγονός που τον καθιστά ιδανικό για τις περιορισμένες επιδόσεις ενός μικροελεγκτή. Ο αλγόριθμος που υλοποιήσαμε ανήκει στη γενική κατηγορία LFSR (Linear Feedback Shift Register) και, πιο συγκεκριμένα, πρόκειται για έναν Fibonacci LFSR.

Αριθμητική αναπαράσταση

Παρουσιάζοντας τη συνάρτηση xnotes, αναφέραμε ότι κάθε νότα περιγράφεται πάντα από δύο bytes. Ωστόσο, η κοινή λογική λέει ότι για να προσδιορίσουμε έναν μουσικό φθόγγο πρέπει να δηλώσουμε νότα, οκτάβα και διάρκεια. Επιπρόσθετα, στην περίπτωση του synthesizer πρέπει να δηλώσουμε το επιθυμητό εφέ και, αν πρόκειται για τετραγωνικό κύμα, το duty cycle. Αναρωτιέστε πώς στριμώξαμε αυτά τα πέντε μεγέθη σε δύο bytes; Θυμηθείτε την περιγραφή των δυνατοτήτων του synthesizer. Τα εφέ που υποστηρίζονται είναι τρία κι αν προσμετρήσουμε την περίπτωση που δεν θέλουμε κανένα εφέ, έχουμε τέσσερεις διαφορετικές καταστάσεις. Προφανώς, για την περιγραφή τεσσάρων καταστάσεων επαρκούν ακριβώς δύο bits. Για το duty cycle γνωρίζουμε ότι το synthesizer υποστηρίζει δύο στάθμες. Επομένως, για τον προσδιορισμό του χρειαζόμαστε μόνο ένα bit. Τέλος, το πλήθος των υποστηριζόμενων αξιών (διαρκειών) ανέρχεται σε οκτώ, που σημαίνει ότι τρία bits αρκούν. Κάντε την πρόσθεση –αν δεν την έχετε κάνει ήδη– και θα διαπιστώσετε ότι όλες αυτές οι πληροφορίες απαιτούν έξι bits και συνεπώς χωράνε άνετα μέσα σε ένα byte. Τώρα, θυμηθείτε τον τρόπο με τον οποίο υπολογίζονται τα διαδοχικά δείγματα ενός κύματος. Για να σχηματίσουμε ένα κύμα συγκεκριμένης συχνότητας, αρκεί να επιλέξουμε μια κατάλληλη τιμή για το μέγεθος phase delta. Στο προηγούμενο άρθρο, αναφέραμε ότι μέσα στο αρχείο melody.asm (γραμμές 8 έως 95) υπάρχει ένας πίνακας που δίνει για κάθε νότα και για κάθε οκτάβα την αντίστοιχη τιμή του phase delta. Επομένως, για να περιγράψουμε νότα και οκτάβα, αρκεί να προσδιορίσουμε ένα στοιχείο του συγκεκριμένου πίνακα. Εφόσον όλες οι νότες και οι παραλλαγές τους είναι μόλις 82, ένα byte φτάνει και περισσεύει.



Το μπλοκ διάγραμμα ενός Linear Feedback Shift Register. Η συνάρτηση ανάδρασης περιλαμβάνει μόνο την πράξη XOR και γι' αυτό λέμε ότι πρόκειται για έναν Fibonacci LFSR. Σε κάθε βήμα εκτελείται η συνάρτηση ανάδρασης, ολισθαίνουν τα bits κατά μία θέση προς τα αριστερά και στη θέση του πρώτου bit τοποθετείται το αποτέλεσμα της πράξης. Η επιλογή των bits (taps) 14 και 15 δεν ήταν τυχαία. Σύμφωνα με ελέγχους που έχουν κάνει πολλοί πριν από εμάς, η χρήση των συγκεκριμένων taps μεγιστοποιεί το πλήθος των παραγόμενων αριθμών.

Τώρα ενδέχεται να αναρωτιέστε γιατί μπήκαμε στον κόπο να συμπυκνώσουμε τόσο πολύ την περιγραφή κάθε νότας. Ο λόγος ήταν απλούστατος: Για να κερδίσουμε χώρο στη μνήμη κάθε καναλιού και, κατ' επέκταση, για να χωράνε μεγαλύτερες μελωδίες. Αν μελετήσετε το περιεχόμενο του αρχείου melody.asm μπορεί να προκύψουν περισσότερα ερωτήματα. Στη γραμμή 148 ορίζουμε μια τιμή για το tempo και μάλιστα εις διπλούν. Από εκεί και κάτω, φαίνεται να έχουμε γράψει μια μελωδία που αξιοποιεί και τα τέσσερα κανάλια. Τι είναι όλ' αυτά; Κατ' αρχάς, αυτές οι τιμές αποτελούν

μέρος του κώδικα και τοποθετούνται στη μνήμη Flash. Γι' αυτό το λόγο άλλωστε χρησιμοποιήσαμε και το αρχικό γράμμα f στα ονόματα των ετικετών :) Όταν ξεκινά το πρόγραμμα, όταν ενεργοποιείται η κάρτα ήχου, εκτελείται η συνάρτηση meminit, η οποία μεταφέρει την τιμή του tempo στο σχετικό καταχωριστή (rhythm) κι αντιγράφει τις νότες από τη μνήμη Flash στις αντίστοιχες περιοχές της μνήμης SRAM. Με αυτό το κολπάκι, κάθε φορά που ενεργοποιείται ο DIY υπολογιστής, η κάρτα ήχου φορτώνει μια προεπιλεγμένη μελωδία. Πρόκειται για τη μελωδία του Bubble Bobble και την τοποθετήσαμε για δύο λόγους: Αφενός επειδή αποτελεί μια καλή επίδειξη των δυνατοτήτων του synthesizer κι αφετέρου επειδή μας αρέσει. Τέλος, η αναφορά του tempo εις διπλούν έχει να κάνει με τον τρόπο που δεσμεύεται η μνήμη Flash. Ο assembler προσπελαύνει τη μνήμη ανά δύο bytes (ανά word των 16bit). Αυτό σημαίνει ότι οι εντολές που δεσμεύουν χώρο για την αποθήκευση αυθαίρετων τιμών (όπως είναι η .db) πρέπει να συνοδεύονται πάντα από άρτιο πλήθος παραμέτρων. Με λίγα λόγια, ήμασταν υποχρεωμένοι να δώσουμε δύο τιμές και αποφασίσαμε να δώσουμε την ίδια εις διπλούν.

Μουσική γραφή

Εσείς προλάβατε την Quick BASIC; Σ' αυτή την εκδοχή της BASIC, όπως και σε αρκετές ακόμα, ο προγραμματιστής μπορούσε να συνθέσει ήχο με τη βοήθεια της play. Η εν λόγω εντολή δεχόταν σαν παράμετρο ένα string, το περιεχόμενο του οποίου περιέγραφε την επιθυμητή μελωδία. Σ' αυτό το string επιτρεπόταν η χρήση συγκεκριμένων χαρακτήρων που ερμηνεύονταν είτε σαν νότες είτε σαν διακόπτες (switches), για την επιλογή οκτάβας, διάρκειας, ρυθμού κ.λπ. Ουσιαστικά, για τη σύνταξη αυτού του string ίσχυαν κάποιοι κανόνες, που στο σύνολό τους αποτελούσαν μια γλώσσα περιγραφής της μουσικής (MML από το Music Macro Language). Αν θέλετε να πάρετε μια γεύση αυτής της γλώσσας, μπορείτε να διαβάσετε την τεκμηρίωση για την εντολή play της Quick BASIC (https://en.wikibooks.org/wiki/QBasic/Appendix#PLAY). Μια τέτοια γλώσσα δημιουργήσαμε κι εμείς για την BASIC που τρέχει στον υπολογιστή μας. Κατ' αρχάς, ορίσαμε τις εντολές mplay και mstop, που είναι πασιφανές τι κάνουν. Επιπρόσθετα, δημιουργήσαμε την εντολή tempo, που δέχεται μια παράμετρο (60, 120, 150 ή 180) και ορίζει το ρυθμό αναπαραγωγής. Το περισσότερο ενδιαφέρον συγκεντρώνεται στην εντολή music, η οποία, όπως και η play, δέχεται σαν παράμετρο ένα ολόκληρο string. Εδώ μπαίνει στο παιχνίδι η MML που επινοήσαμε. Το string έχει πάντα τη μορφή "cmd1 cmd2 ... cmdk", όπου το cmd μπορεί να είναι μία από τις ακόλουθες εντολές:

- Ax: ενεργοποίηση του καναλιού x
- Dx: απενεργοποίηση του καναλιού x
- Cx: διαγραφή του περιεχομένου του καναλιού x
- Mx OND1 OND2 ... ONDk: εισαγωγή στο κανάλι x των μουσικών φθόγγων που περιγράφονται από τις τριάδες OND. Το O δηλώνει την οκτάβα και μπορεί να πάρει τις τιμές από 2 έως 7, ενώ το D δηλώνει την αξία και παίρνει τις

τιμές από 1 (αξία 1/2) έως 8 (αξία 1/64). Το N δηλώνει τη νότα και μπορεί να πάρει μία από τις ακόλουθες τιμές: a, b, c, d, e, f, g, bb, g#, f#, eb και p (για την παύση).

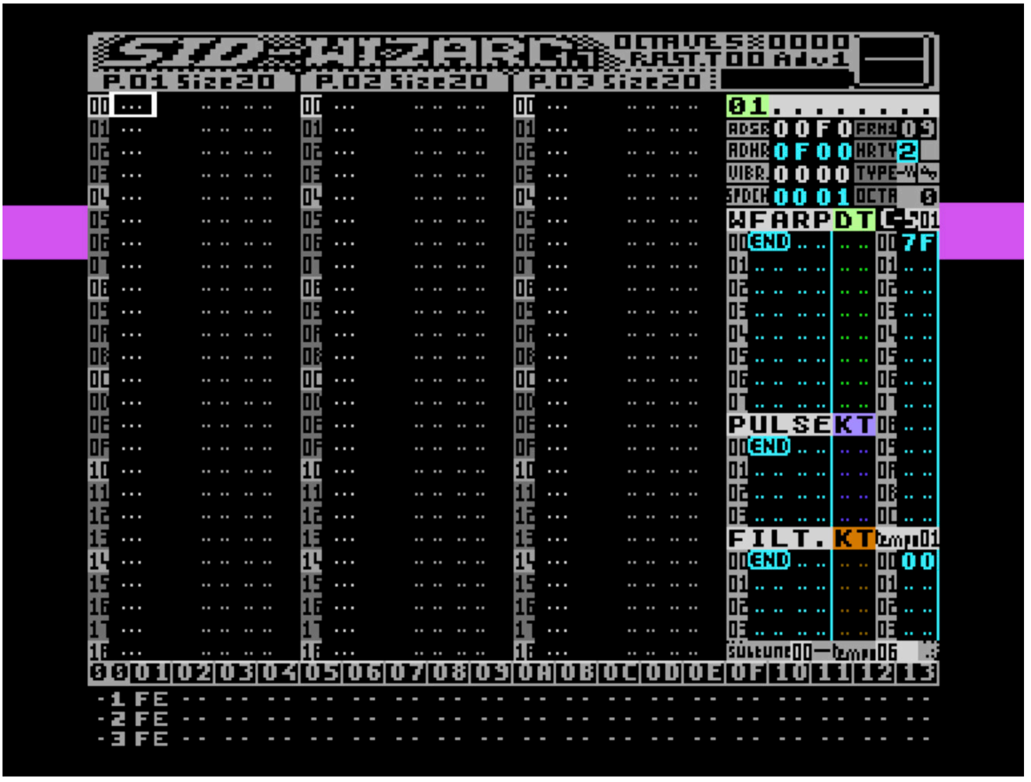
Ειδικά για τις ανάγκες τις εντολής music, κατασκευάσαμε έναν parser μέσα στον ήδη υπάρχοντα parser της BASIC. Όλα αυτά θα τα δούμε αναλυτικότερα σε επόμενο άρθρο, όταν θα παρουσιάσουμε τον κώδικα που τρέχει στον επεξεργαστή του DIY υπολογιστή μας.



Στο πάνω μπλοκ βλέπουμε ένα μικρό απόσπασμα από την Ωδή στη Χαρά (Ode to Joy), γραμμένη στην MML της Quick BASIC. Στο άλλο μπλοκ φαίνεται ίδιο μουσικό απόσπασμα γραμμένο με τη δική μας MML, στη BASIC που τρέχει ο DIY υπολογιστής μας. Στη γραμμή 20 της δεύτερης εκδοχής, πριν αρχίσουμε να προσθέτουμε νότες, φροντίζουμε να απενεργοποιήσουμε τα κανάλια 2, 3 και 4, όπως επίσης και να καθαρίσουμε το 1. Έτσι είναι... Όσο πιο πλούσιο synthesizer έχεις, τόσο αυξάνονται και οι ευθύνες ;)

Πίσω στο (μεγάλο) PC

Μ' αυτά και με τ' άλλα, η κάρτα ήχου ολοκληρώθηκε κι εμείς είχαμε μείνει να την κοιτάμε. Είχαν προηγηθεί τόσες πολλές δοκιμές, που ήμασταν απολύτως βέβαιοι ότι δουλεύει άψογα. Παρ' όλα αυτά, μην έχοντας τις απαραίτητες μουσικές γνώσεις, ήταν αδύνατο να συνθέσουμε κάτι ενδιαφέρον. Μόνο μερικές απλοϊκές, μονότονες και φάλτσες μελωδίες καταφέρναμε να γράψουμε. Κάπως έτσι αποφασίσαμε να ζητήσουμε τη βοήθεια ενός φίλου μουσικού. Τι θα του λέγαμε όμως; Θα του ζητούσαμε να μάθει την MML ή μήπως θα του δίναμε τον DIY υπολογιστή, για να κάνει δοκιμές γράφοντας προγράμματα σε BASIC; Παρόμοια προβλήματα αντιμετώπιζαν και οι κατασκευαστές των arcade games. Η μουσική γι' αυτά τα παιχνίδια αποθηκευόταν σε κάποια ROM και, όπως αντιλαμβάνεστε, η εγγραφή απαιτούσε εξειδικευμένα εργαλεία. Ένας μουσικός θα ήταν πρακτικά αδύνατο να κάνει τη δουλειά του, αν για κάθε δοκιμή έπρεπε να προγραμματίζει κάποιο τσιπάκι. Για να διευκολυνθεί αυτή η εργασία, οι προγραμματιστές της εποχής ανέπτυξαν τους λεγόμενους trackers. Πρόκειται για προγράμματα που έτρεχαν σε κανονικούς υπολογιστές και εξομοίωναν πλήρως τα synthesizers των arcades. Ε, λοιπόν, με τον ίδιο τρόπο λύσαμε και το δικό μας πρόβλημα: Κατασκευάσαμε έναν tracker που εξομοιώνει πλήρως το synthesizer της κάρτας ήχου και μπορεί να τρέξει σε οποιοδήποτε σύγχρονο PC. Για την ανάπτυξη του tracker επιλέξαμε τη C#. Έχοντας περάσει μερικούς μήνες αναπτύσσοντας το firmware της κάρτας ήχου σε Assembly, είναι αδύνατο να περιγράψουμε τον ενθουσιασμό που νιώθαμε, στη σκέψη ότι θα χρησιμοποιούσαμε μια τόσο μοντέρνα γλώσσα :D Μπορείτε να κατεβάσετε το πακέτο με το project από το http://bit.ly/dh053dsynth.



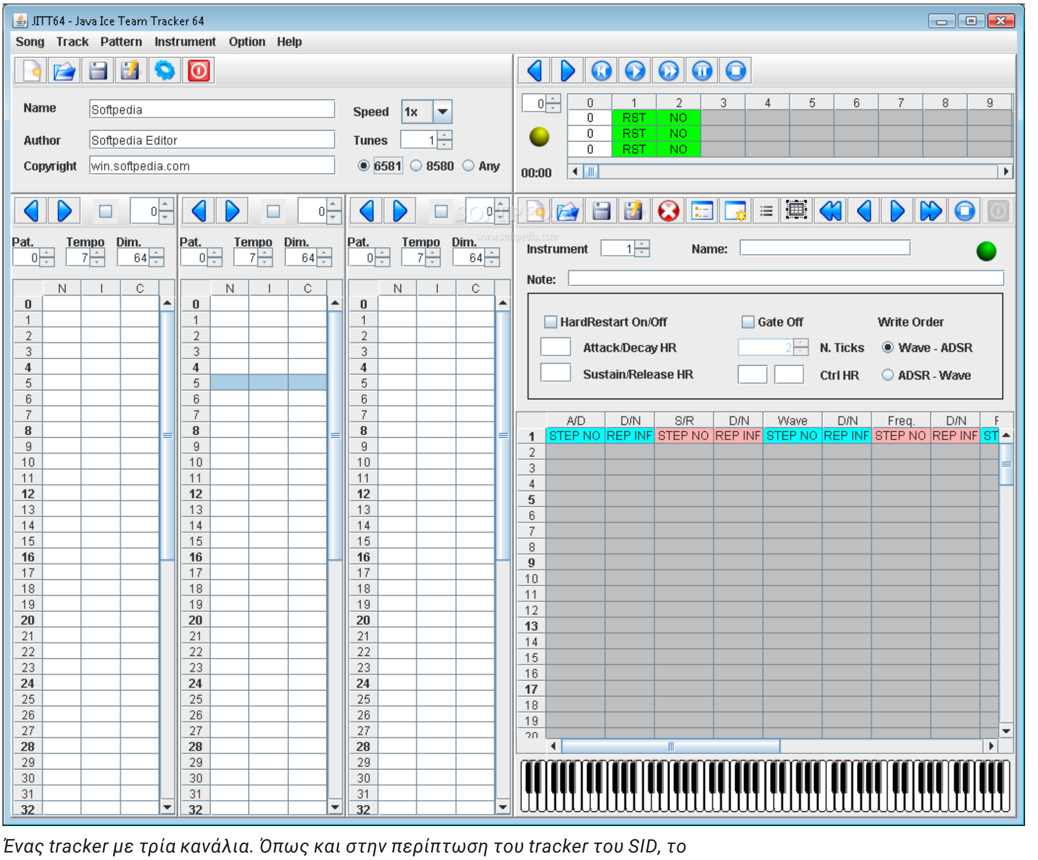
Περιμένατε ότι δεν θα υπήρχε; Στην εικόνα φαίνεται ένας tracker για το SID! Πρόκειται για ένα πολύ παλιό πρόγραμμα, που επέτρεπε στους χρήστες του να εξομοιώνουν ένα ακόμα παλιότερο τσιπάκι ήχου.

Το τίμημα της καθαρότητας

Για την κατασκευή του GUI ακολουθήσαμε μια ανορθόδοξη προσέγγιση. Αρχικά χρησιμοποιήσαμε το εργαλείο σχεδίασης που προσφέρει το Visual Studio. Έτσι προέκυψε το αρχείο Form1.Designer.cs, που περιλάμβανε τις δηλώσεις και τις ιδιότητες για τα αντικείμενα (text box, list box, button κ.ά.) που συνθέτουν το περιβάλλον. Στη συνέχεια όμως τροποποιήσαμε το περιεχόμενο του αρχείου με τέτοιο τρόπο, που ο Designer του Visual Studio δεν αναγνώριζε το περιεχόμενο. Καταλαβαίνετε τι σημαίνει αυτό; Αν ανοίξετε το πρόγραμμά μας με οποιοδήποτε IDE, θα διαπιστώσετε ότι δεν μπορεί να σχεδιάσει το GUI. Επομένως, είναι αδύνατο να το τροποποιήσετε χρησιμοποιώντας κάποιο visual εργαλείο. Παρ' όλα αυτά το πρόγραμμα εκτελείται κανονικά. Γι' αυτή την αναποδιά ευθύνεται η όρεξή μας για πειράματα.

Το περιβάλλον του προγράμματος μοιάζει με εκείνο των περισσοτέρων trackers: Για κάθε κανάλι του synthesizer, περιλαμβάνει ένα σύνολο από controls που επιτρέπουν στο χρήστη να προσθαφαιρεί και να τροποποιεί νότες. Αυτό σημαίνει ότι τα περισσότερα στοιχεία του περιβάλλοντος επαναλαμβάνονται: Υπάρχουν τέσσερα κουμπιά για την προσθήκη μιας νότας, τέσσερα combos για την επιλογή οκτάβας, άλλα τέσσερα για την επιλογή της διάρκειας και πάει λέγοντας. Ακριβώς αυτή η επανάληψη ήταν που μας ώθησε να επέμβουμε τόσο βάρβαρα στο form1.Designer.

cs. Αποφασίσαμε να ομαδοποιήσουμε τα controls με την ίδια λειτουργία και να τα τοποθετήσουμε μέσα σε πίνακες των τεσσάρων στοιχείων (όσα είναι και τα κανάλια). Για παράδειγμα, όλα τα κουμπιά για την προσθήκη μιας νότας τοποθετήθηκαν στον πίνακα insert_note, ενώ όλα τα κουμπιά για την τροποποίηση μιας νότας μπήκαν στον πίνακα edit_note. Αναρωτιέστε τι πετύχαμε με αυτόν τον τρόπο; Σκεφτείτε το εξής: Ας υποθέσουμε ότι έχουμε γράψει μια γενική συνάρτηση, που μπορεί προσθέτει νότες στη λίστα ενός καναλιού. Σε κάθε κλήση της εν λόγω συνάρτησης θα έπρεπε να δίνουμε τουλάχιστον πέντε παραμέτρους, με τα αντικείμενα του επιλεγμένου καναλιού: Τέσσερα combo boxes (για τη νότα, την οκτάβα, τη διάρκεια και το εφέ) κι ένα list box (για την αποθήκευση της νότας). Με τη χρήση των πινάκων, η ίδια συνάρτηση μπορεί να λειτουργεί με μία μόνο παράμετρο: τον αριθμό του καναλιού. Τελικά, με τη βοήθεια των πινάκων καταφέραμε να γράψουμε πολύ πιο καθαρό κώδικα ή, τέλος πάντων, κώδικα που διαβάζεται ευκολότερα από εμάς. Πάντως, το τίμημα του να μη δουλεύουν οι GUI designers είναι αρκετά υψηλό και για να 'μαστε ειλικρινείς είναι αμφίβολο αν θα ακολουθούσαμε ξανά την ίδια προσέγγιση.



Ένας tracker με τρία κανάλια. Όπως και στην περίπτωση του tracker του SID, το μεγαλύτερο τμήμα του περιβάλλοντος καταλαμβάνεται από τρεις λίστες. Σε αυτές τοποθετούνται οι νότες που θα παίξει κάθε κανάλι. (Screenshot source: Softpedia.)

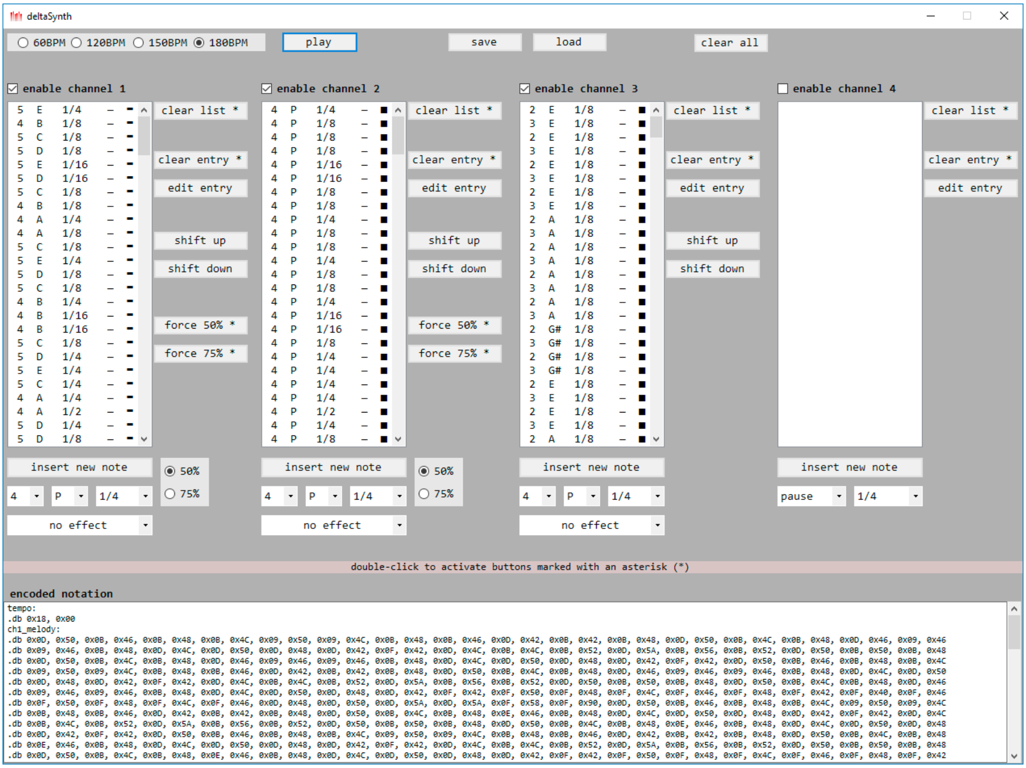
Μηχανισμός τύχης

Οι γεννήτριες LFSR στηρίζονται σ' έναν καταχωριστή, του οποίου τα ψηφία (bits) ολισθαίνουν διαρκώς προς μια κατεύθυνση. Όπως αντιλαμβάνεστε, κάθε φορά που πραγματοποιείται η ολίσθηση απομακρύνεται ένα bit από τον καταχωριστή και, ταυτόχρονα, δημιουργείται μια κενή θέση. Για παράδειγμα, αν η ολίσθηση έχει κατεύθυνση προς τα' αριστερά, σε κάθε βήμα της διαδικασίας προκύπτει ένα κενό στο δεξιό άκρο του καταχωριστή. Εκεί τοποθετείται ένα ψηφίο, που προκύπτει από μια γραμμική συνάρτηση κάποιων άλλων bits του καταχωριστή. Η εν λόγω συνάρτηση ονομάζεται συνάρτηση ανάδρασης και μπορεί να αξιοποιεί όσα και όποια bits επιθυμούμε. Η επιλογή των bits (ονομάζονται taps) παίζει καθοριστικό ρόλο για την συμπεριφορά του LFSR. Σκεφτείτε το εξής: Ένας 16μπιτος καταχωριστής μπορεί να πάρει 2^16 (=65536) διαφορετικές τιμές. Παρ' όλα αυτά, δεν είναι βέβαιο ότι *κάθε* 16μπιτος LFSR θα παράγει τόσες διαφορετικές τιμές. Επιλέγοντας κάποια taps μπορούμε να δημιουργήσουμε μια "φτωχή" εκδοχή, που θα παράγει και θα ανακυκλώνει μερικές μόνο από τις 65536 τιμές. Εξίσου δυνατή είναι και η κατασκευή ενός maximal LFSR, που θα παράγει όλους τους δυνατούς αριθμούς. Σε γενικές γραμμές, η επιλογή των σωστών taps δεν αποτελεί εύκολη υπόθεση. Είναι αδύνατο να προβλέψουμε τη συμπεριφορά ενός LFSR, αν δεν υπολογίσουμε μία προς μία τις τιμές που παράγει. Το πρόβλημα της επιλογής των taps λύνεται πανεύκολα μόνο για τους Fibonacci LFSR. Έτσι ονομάζονται οι LFSR των οποίων η συνάρτηση ανάδρασης αποτελείται μόνο από την πράξη XOR. Όπως υποψιάζεστε, πρόκειται για το δημοφιλέστερο είδος LFSR και είναι πολλοί εκείνοι που έχουν μελετήσει τη συμπεριφορά τους διεξοδικά. Το Internet είναι γεμάτο με λίστες που αναφέρουν τα κατάλληλα taps, για την κατασκευή maximal LFSR με οποιοδήποτε πλήθος bits.

Παίξτε κι εσείς!

Ο κώδικας για τον tracker, αν προσπεράσουμε τις εκτεταμένες επεμβάσεις στο Form1.Designer.cs, εκτείνεται στα αρχεία Form1.cs και synthesizer.cs. Όπως υποψιάζεστε, το πρώτο αρχείο περιλαμβάνει τις συναρτήσεις για το χειρισμό των διαφόρων events του GUI. Το περισσότερο ενδιαφέρον συγκεντρώνεται στο synthesizer.cs, όπου ορίζουμε την κλάση με το (πολύ πρωτότυπο) όνομα synthesizer. Οι μέθοδοι αυτής της κλάσης σχηματίζουν τα κύματα που περιγράφουν οι νότες, υπολογίζοντας ένα προς ένα τα διαδοχικά δείγματα. Ακολούθως, για την αναπαραγωγή των δειγμάτων τοποθετούνται σε ένα αρχείο WAV. Για τον υπολογισμό των δειγμάτων δεν χρειάζεται να πούμε κάτι. Αν έχετε κατανοήσει τον αντίστοιχο μηχανισμό στην Assembly, ο κώδικας σε C# θα σας φανεί περίπατος. Σε κάθε περίπτωση, έχετε κατά νου ότι ο tracker γράφτηκε πολύ βιαστικά και θα ήταν φρόνιμο να υιοθετήσετε τις πρακτικές που ακολουθήσαμε κατά την ανάπτυξή του. Ο μόνος λόγος για τον οποίο σας δίνουμε τον tracker, είναι για να μπορείτε να παίξετε κι εσείς με τους όμορφους, χοντροκομμένους ήχους που βγάζει η κάρτα μας. Κι αν όλα όσα έχετε διαβάσει δεν κατάφεραν να σας ανοίξουν την όρεξη, μπορείτε να κατεβάσετε αυτό το πακέτο από το http://bit.ly/dh053tunes, με δυο διάσημες μελωδίες. Πρόκειται για την εισα-

γωγή του Bubble Bobble και του Tetris, γραμμένες στον tracker και αποθηκευμένες σε επεξεργάσιμη μορφή. Μπορείτε να τις φορτώσετε, να τις ακούσετε και να τις τροποποιήσετε όσο θέλετε.



Ο δικός μας tracker! Όταν πατάμε το κουμπί play, στην κάτω περιοχή του παραθύρου εμφανίζεται η μουσική με τη μορφή που θα αποθηκευτεί στον μικροελεγκτή της κάρτας ήχου. Αν πάρουμε αυτό το κατεβατό και το τοποθετήσουμε στο αρχείο melody.asm, στη θέση της μελωδίας του Bubble Bobble, θα έχουμε αλλάξει την εργοστασιακή μελωδία που συνοδεύει την κάρτα ήχου. Παρεμπιπτόντως, στην εικόνα φαίνεται η μελωδία του Tetris :)

Skill: Beginner
Tags: Jabber, XMPP, Openfire, OTR, Active Directory, Pidgin, Spark

Κάντε το chat πραγματικά προσωπική υπόθεση

Πόσο συχνά επικοινωνείτε μέσω Διαδικτύου με χρήση δημοφιλών υπηρεσιών IM; Γνωρίζετε ότι οι εταιρείες πίσω από τις εν λόγω υπηρεσίες δηλώνουν πως, αν χρειαστεί, θα συνεργαστούν με τις τοπικές αρχές για την άρση του απορρήτου σας; Πόσο εμπιστεύεστε τις εταιρείες που υπόσχονται ότι ποτέ δεν θα έκαναν κάτι τέτοιο;

του Πέτρου Κυλαδίτη

Δεν ξέρουμε πώς το βλέπετε εσείς, εμείς όμως που παίρνουμε την ασφάλεια των προσωπικών μας δεδομένων στα σοβαρά αποφασίσαμε να στήσουμε τον δικό μας chat server. Συγκεκριμένα, καταφεύγοντας σε ανοιχτά πρωτόκολλα, θα υλοποιήσουμε την υπηρεσία chat στο πλαίσιο του τοπικού μας δικτύου.

Μια κατηγορία ιντερνετικών υπηρεσιών που έχουμε χρησιμοποιήσει όλοι, είναι το Instant Messaging ή απλά IM ή ακόμα πιο απλά σκέτο chat. Η αρχή έγινε με το IRC, που τέθηκε στο περιθώριο με την ανάπτυξη των σύγχρονων δικτύων IM. Τα δίκτυα του είδους προσφέρουν περισσότερες ευκολίες από το IRC και μια πιο προσωποποιημένη προσέγγιση στις συζητήσεις. Σε αυτά προστέθηκαν και τα δίκτυα κοινωνικής δικτύωσης, που το καθένα προσφέρει ένα δικό του μέσο επικοινωνίας κι άμεσης ανταλλαγής μηνυμάτων. Παρ' όλη την εξέλιξη, αν και η ανταλλαγή μηνυμάτων ξέφυγε από τον απρόσωπο και μαζικό χαρακτήρα του IRC, η επικοινωνία δεν έγινε περισσότερο ιδιωτική, ούτε βέβαια εξασφαλίστηκε το απόρρητο. Οι εταιρείες που διαχειρίζονται τα διάφορα δίκτυα IM δηλώνουν ότι μπορούν να ελέγχουν και να καταγράφουν τα μηνύματα για λόγους διαφημιστικής προώθησης, προστασίας πνευματικών δικαιωμάτων, εθνικής ασφάλειας κ.ά. Εμείς δεν έχουμε φοβερά μυστικά να κρύψουμε, αλλά η ιδέα ότι κάποιοι μπορούν να κάνουν ό,τι θέλουν με τα μηνύματά μας καθόλου δεν μας αρέσει.

Ο κόσμος του Jabber

Σε πείσμα των καιρών, αποφασίσαμε να στήσουμε το δικό μας chat server, σε ένα από τα σχεδόν-ευφυώς κατανεμημένα ανά την επικράτεια εργαστήρια της Parabing Creations. Για την υλοποίηση του project επιλέξαμε να χρησιμοποιήσουμε το πρωτόκολλο XMPP, που είναι γνωστό και σαν Jabber. Το επίσημο όνομα του πρωτοκόλλου προέρχεται από τις λέξεις Extensible Messaging and Presence Protocol. Το δεύτερο όνομα, το Jabber, είναι εκείνο που είχε επιλέξει ο δημιουργός του, ο Jeremie Miller, όταν το πρωτοέφτιαξε το 1998. Το 2004 το Jabber μετατράπηκε σε πρότυπο από τον οργανισμό IEFT, μετονομάστηκε σε XMPP και η διαχείρισή του πέρασε στο ίδρυμα Jabber Software Foundation.

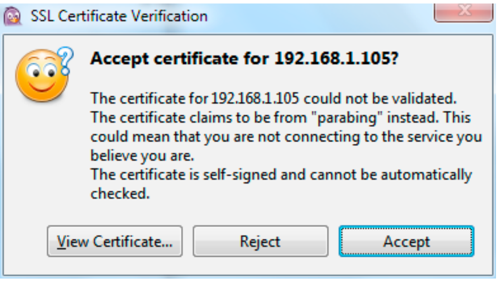
Αν και για τη λειτουργία του Jabber απαιτείται η ύπαρξη ενός XMPP server, θεωρείται αποκεντρωμένο σύστημα επικοινωνίας. Αυτό συμβαίνει διότι οποιοσδήποτε το επιθυμεί, μπορεί να στήσει τον δικό του XMPP server. Οι χρήστες που συνδέονται στον ίδιο server επικοινωνούν χωρίς τη μεσολάβηση κάποιου ξένου μηχανήματος. Οι χρήστες διαφορετικών servers, ανταλλάσσουν μηνύματα μέσω αυτών. Αναρωτιέστε που κολλάει ο όρος "αποκεντρωμένο"; Το δίκτυο Jabber δεν είναι αποκεντρωμένο στο επίπεδο των χρηστών, όπως συμβαίνει στα δίκτυα P2P. Είναι όμως αποκεντρωμένο στο επίπεδο των servers. Οι Jabber servers μπορούν να συνδέουν μεταξύ τους ολόκληρα δίκτυα και, υπό μία έννοια, λειτουργούν ως gateways. Με απλά λόγια, ο χρήστης ενός Jabber server μπορεί να επικοινωνήσει με χρήστες άλλων Jabber servers, όπως και με χρήστες του Google Talk, του Facebook Messenger κ.ά. Μάλιστα, με τη μεσολάβηση των κατάλληλων Jabber servers, ένας χρήστης μπορεί να στείλει ακόμα και email ή SMS.

Ένα ακόμα στοιχείο που αναδεικνύει τις πλούσιες δυνατότητες του δικτύου Jabber σχετίζεται με τη δομή που έχουν οι ταυτότητες των χρηστών. Κάθε χρήστης του Jabber λαμβάνει ένα μοναδικό αναγνωριστικό, που ονομάζεται Jabber ID ή απλά JID. Αυτό το αναγνωριστικό έχει τη μορφή user@server/resource. Προφανώς, user είναι το όνομα του χρήστη και server είναι το όνομα του Jabber server, στον οποίο είναι συνδεδεμένος ο χρήστης. Το resource αποτελεί ένα γενικό αναγνωριστικό της τοποθεσίας από την οποία συνδέεται κάθε φορά ο χρήστης. Έτσι, το resource μπορεί να πάρει τιμές όπως mobile, work ή home, ενώ μπορεί κάλλιστα να δανειστεί και το όνομα του client. Εξαιτίας της δομής που έχουν τα JID, ο εκάστοτε αποστολέας μπορεί να στέλνει μηνύματα σε συγκεκριμένο σημείο σύνδεσης ενός χρήστη ή και σε όλα. Για παράδειγμα, μπορεί να στείλει ένα μήνυμα στο γραφείο του Γιάννη ή σε όλες τις θέσεις από τις οποίες συνδέεται ο Γιάννης. Στην πρώτη περίπτωση, ο υποθετικός συνομιλητής θα πάρει το μήνυμα μόνο όταν συνδεθεί από το γραφείο.

Το Jabber υποστηρίζει εγγενώς την ασφαλή επικοινωνία μεταξύ server και χρήστη. Δυστυχώς αυτό δεν ισχύει για όλη τη διαδρομή των μηνυμάτων, από τον ένα χρήστη μέχρι τον επόμενο. Η διασφάλιση της προστασίας της επικοινωνίας μεταξύ δύο χρηστών, επαφίεται στους ενδιάμεσους servers και στους αντίστοιχους διαχειριστές. Εμείς θα στήσουμε τον δικό μας server και σε ό,τι αφορά στην τοπική ανταλλαγή μηνυμάτων δεν θα έχουμε τίποτα να φοβηθούμε. Από κει και πέρα, έχουμε την

επιλογή να στραφούμε στο πρωτόκολλο Off-the-Record Messaging (OTR). Μπορείτε να το φαντάζεστε ως ένα layer κρυπτογράφησης πάνω στο Jabber. Το OTR υποστηρίζεται από αρκετούς clients.

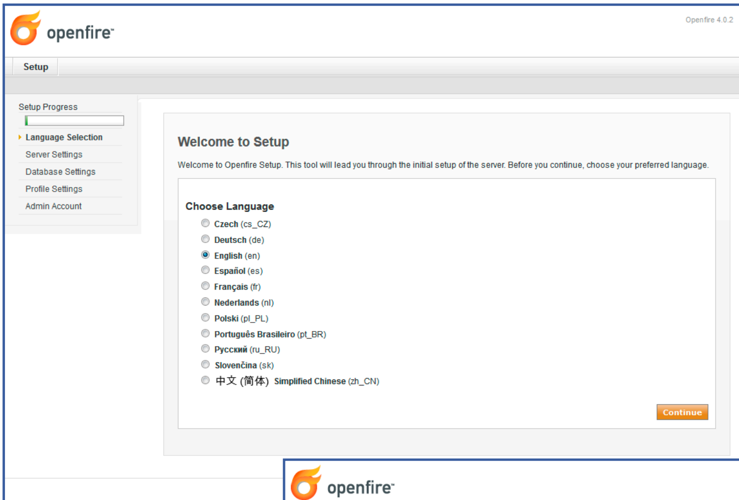
Οι επιλογές ανάμεσα στους διαθέσιμους Jabber servers είναι πολλές. Εμείς στραφήκαμε στον OpenFire: αποτελεί προϊόν Ανοιχτού Κώδικα, είναι γραμμένος σε Java και διατίθεται υπό την άδεια χρήσης Apache. Μπορείτε να τον κατεβάσετε από το www.igniterealtime.org/projects/openfire. Ο OpenFire υποστηρίζει όλες τις λειτουργίες του πρωτοκόλλου Jabber, ενώ δέχεται και διάφορα extensions που επεκτείνουν τις λειτουργίες του περεταίρω. Η πιστοποίηση των χρηστών πραγματοποιείται με δύο τρόπους: είτε με το απλό μοντέλο της λίστας χρηστών, σε μια βάση δεδομένων, είτε με τη συνδρομή της υπηρεσίας LDAP. Ακριβώς αυτή την υπηρεσία προσφέρει και ο domain controller που στήσαμε για το εργαστήριό μας, στο τεύχος 051. Όπως αντιλαμβάνεστε, δεν σκοπεύουμε να τον αφήσουμε αναξιοποίητο ;)

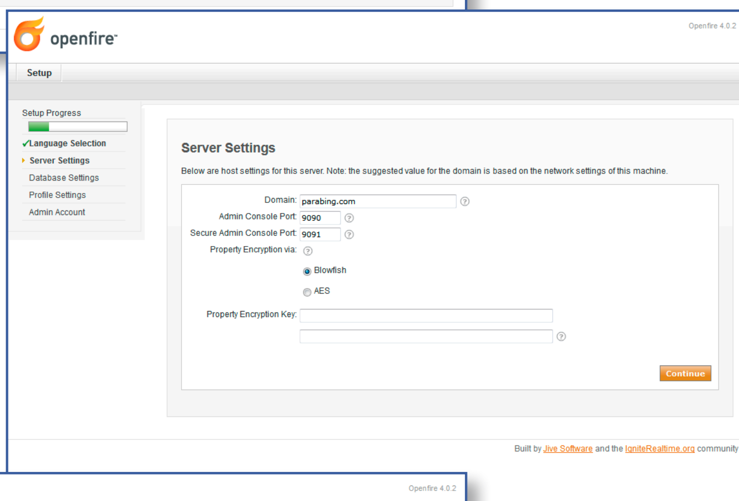


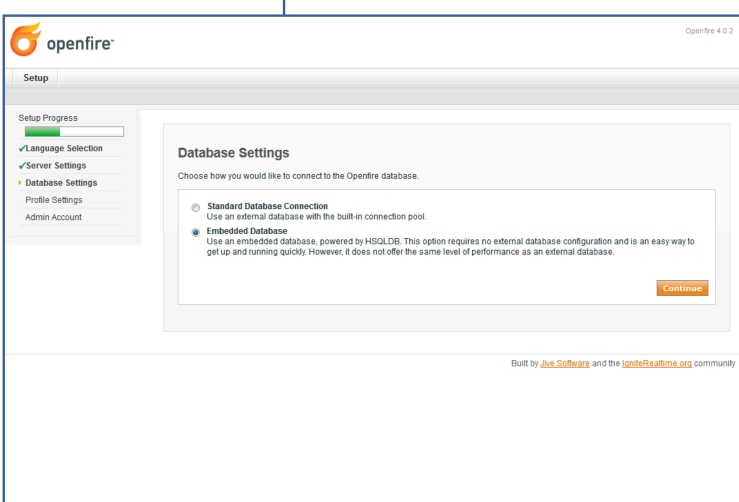
Το πρωτόκολλο XMPP υποστηρίζει την ασφαλή επικοινωνία μεταξύ server και client. Από κει και πέρα, τα πάντα εξαρτώνται από τον διαχειριστή του server. Αν δεν υπάρχει εμπιστοσύνη, μπορούμε να στραφούμε στη λύση του OTR.

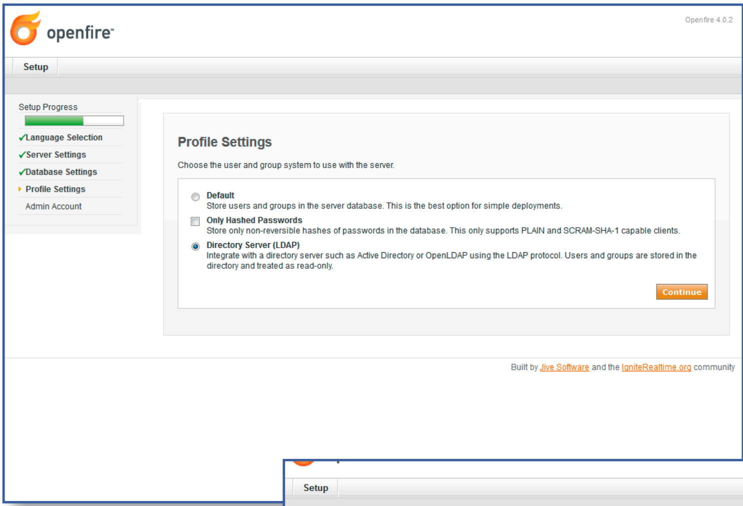
Εγκατάσταση και ρύθμιση

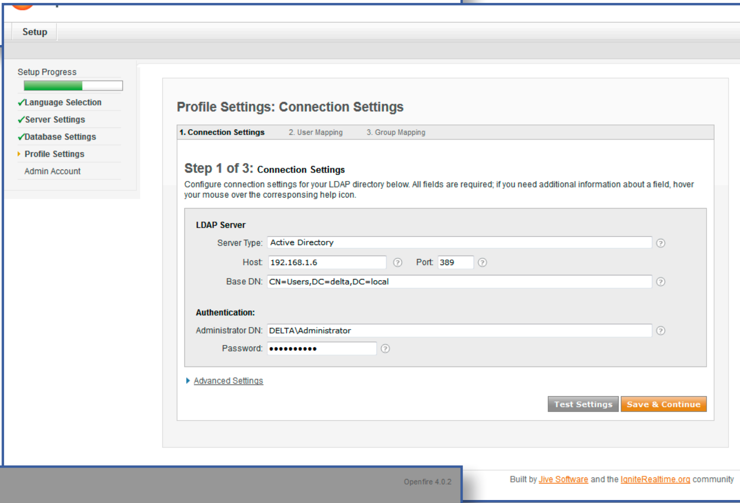
Η εργασία μας ξεκίνησε με τη λήψη του πακέτου του OpenFire για Windows. Άσχετα με την τελική χρήση που είχαμε κατά νου, το εγκαταστήσαμε σ' έναν υπολογιστή-αρχαιολογικό κειμήλιο, που μέχρι πρότινος καθόταν σε μια γωνιά του εργαστηρίου. Αυτή η επιλογή μοιάζει τουλάχιστον παράξενη, αλλά θέλαμε να δοκιμάσουμε την απόδοση και τη σταθερότητα του server σ' ένα μηχάνημα περιορισμένων δυνατοτήτων. Η εγκατάσταση του OpenFire πραγματοποιήθηκε γρήγορα κι αμέσως μετά εμφανίστηκε ένα παράθυρο με τέσσερα κουμπιά και την κονσόλα μηνυμάτων (logs). Τα κουμπιά προσφέρουν βασικές λειτουργίες (έναρξη, διακοπή, τερματισμός και ρύθμιση του server), αλλά δεν είμαστε υποχρεωμένοι να τα χρησιμοποιήσουμε. Ο OpenFire διαθέτει κι ένα web interface, που επιτρέπει την εύκολη ρύθμιση από οποιοδήποτε άλλο μηχάνημα του τοπικού δικτύου. Έτσι, εγκαταλείψαμε το δυσκίνητο "σαπάκι" και μεταφερθήκαμε στον κανονικό υπολογιστή μας. Από εκεί συνδεθήκαμε στη διεύθυνση του προηγούμενου μηχανήματος και συγκεκριμένα στην προεπιλεγμένη θύρα που ακούει το web interface (9090). Κάπως έτσι, επειδή ο OpenFire ήταν ακόμα αρρύθμιστος, βρεθήκαμε μπροστά σε έναν οδηγό ρύθμισης.

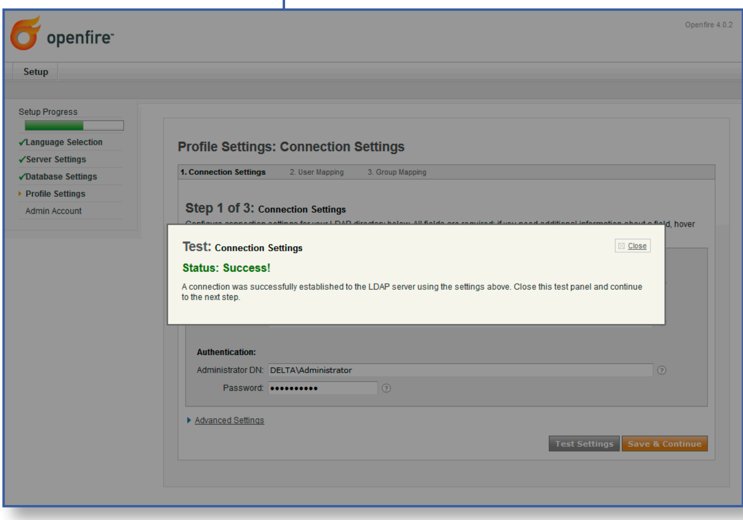

openfire
Openfire 4.0.2
Setup
Setup Progress
Language Selection
Server Settings
Database Settings
Profile Settings
Admin Account
Welcome to Setup
Welcome to Openfire Setup. This tool will lead you through the initial setup of the server. Before you continue, choose your preferred language.
Choose Language
Czech (cs_CZ)
Deutsch (de)
English (en)
Español (es)
Français (fr)
Nederlands (nl)
Polski (pl_PL)
Português Brasileiro (pt_BR)
Русский (ru_RU)
Slovenčina (sk)
中文 (简体) Simplified Chinese (zh_CN)
Continue

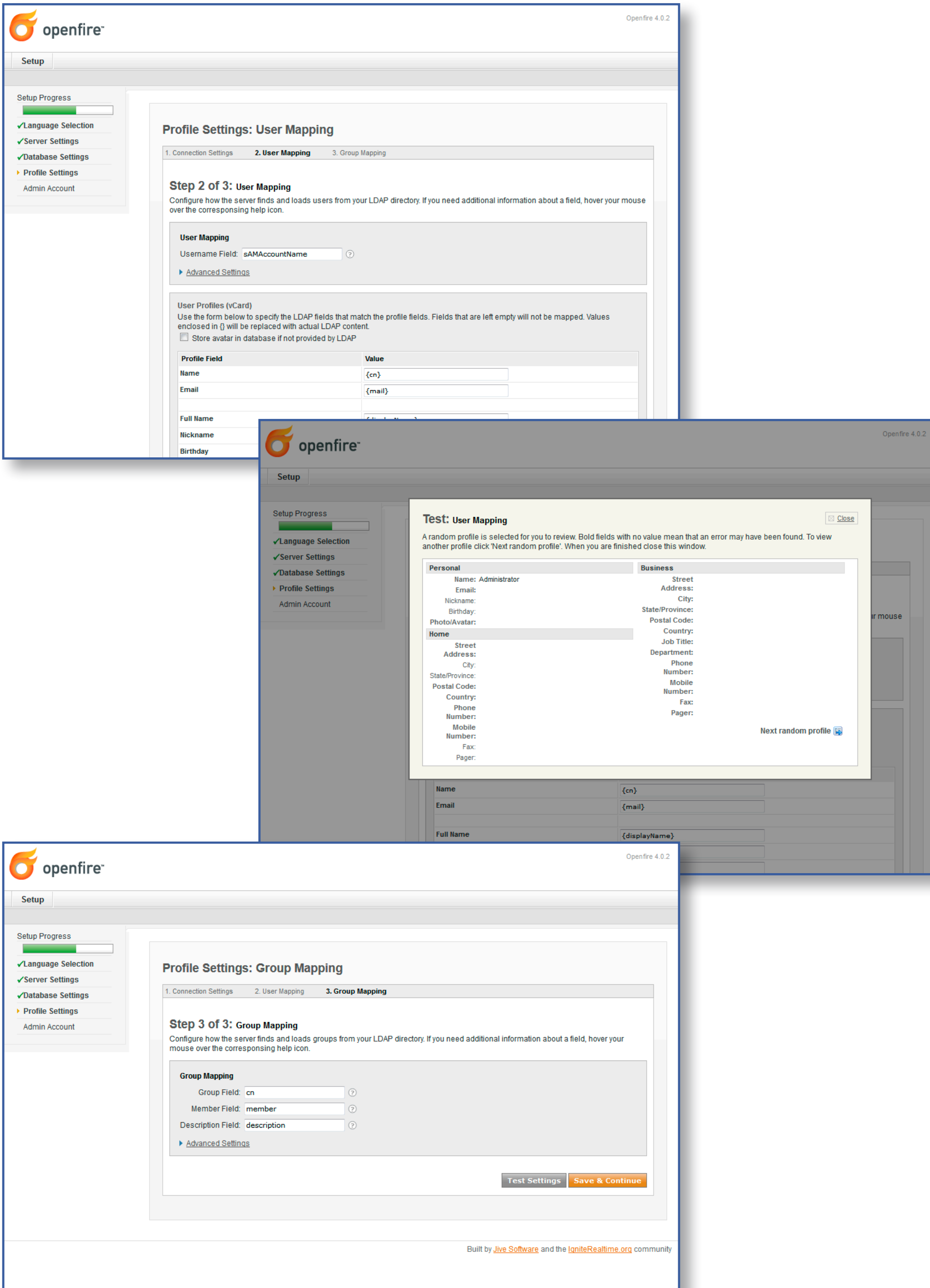

openfire
Openfire 4.0.2
Setup
Setup Progress
Language Selection
Server Settings
Database Settings
Profile Settings
Admin Account
Server Settings
Below are host settings for this server. Note: the suggested value for the domain is based on the network settings of this machine.
Domain: parabing.com
Admin Console Port: 9090
Secure Admin Console Port: 9091
Property Encryption via:
Blowfish
AES
Property Encryption Key:
Continue
Built by Jive Software and the IgniteRealtime.org community

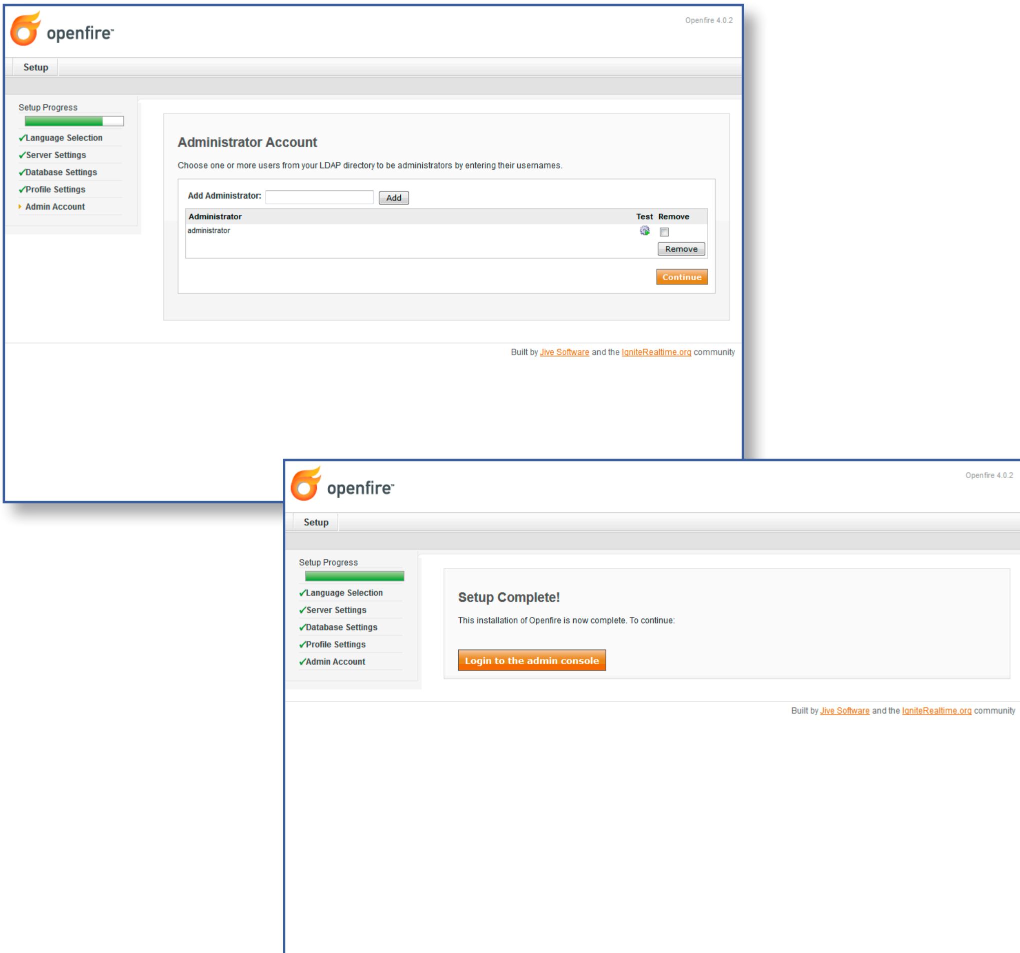

openfire
Openfire 4.0.2
Setup
Setup Progress
Language Selection
Server Settings
Database Settings
Profile Settings
Admin Account
Database Settings
Choose how you would like to connect to the Openfire database.
Standard Database Connection
Use an external database with the built-in connection pool.
Embedded Database
Use an embedded database, powered by HSQLDB. This option requires no external database configuration and is an easy way to get up and running quickly. However, it does not offer the same level of performance as an external database.
Continue
Built by Jive Software and the IgniteRealtime.org community


openfire
Openfire 4.0.2
Setup
Setup Progress
Language Selection
Server Settings
Database Settings
Profile Settings
Admin Account
Profile Settings
Choose the user and group system to use with the server.
Default
Store users and groups in the server database. This is the best option for simple deployments.
Only Hashed Passwords
Store only non-reversible hashes of passwords in the database. This only supports PLAIN and SCRAM-SHA-1 capable clients.
Directory Server (LDAP)
Integrate with a directory server such as Active Directory or OpenLDAP using the LDAP protocol. Users and groups are stored in the directory and treated as read-only.
Continue
Built by Jive Software and the IgniteRealtime.org community


Setup
Setup Progress
Language Selection
Server Settings
Database Settings
Profile Settings
Admin Account
Profile Settings: Connection Settings
1. Connection Settings
2. User Mapping
3. Group Mapping
Step 1 of 3: Connection Settings
Configure connection settings for your LDAP directory below. All fields are required; if you need additional information about a field, hover your mouse over the corresponding help icon.
LDAP Server
Server Type: Active Directory
Host: 192.168.1.6
Port: 389
Base DN: CN=Users,DC=delta,DC=local
Authentication:
Administrator DN: DELTA\Administrator
Password: ••••••••••
Advanced Settings
Test Settings
Save & Continue
Built by Jive Software and the IgniteRealtime.org community


openfire
Openfire 4.0.2
Setup
Setup Progress
Language Selection
Server Settings
Database Settings
Profile Settings
Admin Account
Profile Settings: Connection Settings
1. Connection Settings
2. User Mapping
3. Group Mapping
Step 1 of 3: Connection Settings
Test: Connection Settings
Close
Status: Success!
A connection was successfully established to the LDAP server using the settings above. Close this test panel and continue to the next step.
Authentication:
Administrator DN: DELTA\Administrator
Password: ••••••••••
Advanced Settings
Test Settings
Save & Continue


openfire
Openfire 4.0.2
Setup
Setup Progress
Language Selection
Server Settings
Database Settings
Profile Settings
Admin Account
Profile Settings: User Mapping
1. Connection Settings
2. User Mapping
3. Group Mapping
Step 2 of 3: User Mapping
Configure how the server finds and loads users from your LDAP directory. If you need additional information about a field, hover your mouse over the corresponding help icon.
User Mapping
Username Field: sAMAccountName
Advanced Settings
User Profiles (vCard)
Use the form below to specify the LDAP fields that match the profile fields. Fields that are left empty will not be mapped. Values enclosed in {} will be replaced with actual LDAP content.
Store avatar in database if not provided by LDAP
Profile Field
Value
Name
{cn}
Email
{mail}
Full Name
Nickname
Birthday
Test: User Mapping
Close
A random profile is selected for you to review. Bold fields with no value mean that an error may have been found. To view another profile click 'Next random profile'. When you are finished close this window.
Personal
Name: Administrator
Email:
Nickname:
Birthday:
Photo/Avatar:
Home
Street Address:
City:
State/Province:
Postal Code:
Country:
Phone Number:
Mobile Number:
Fax:
Pager:
Business
Street Address:
City:
State/Province:
Postal Code:
Country:
Job Title:
Department:
Phone Number:
Mobile Number:
Fax:
Pager:
Next random profile
{displayName}
Profile Settings: Group Mapping
Step 3 of 3: Group Mapping
Configure how the server finds and loads groups from your LDAP directory. If you need additional information about a field, hover your mouse over the corresponding help icon.
Group Mapping
Group Field: cn
Member Field: member
Description Field: description
Advanced Settings
Test Settings
Save & Continue
Built by Jive Software and the IgniteRealtime.org community

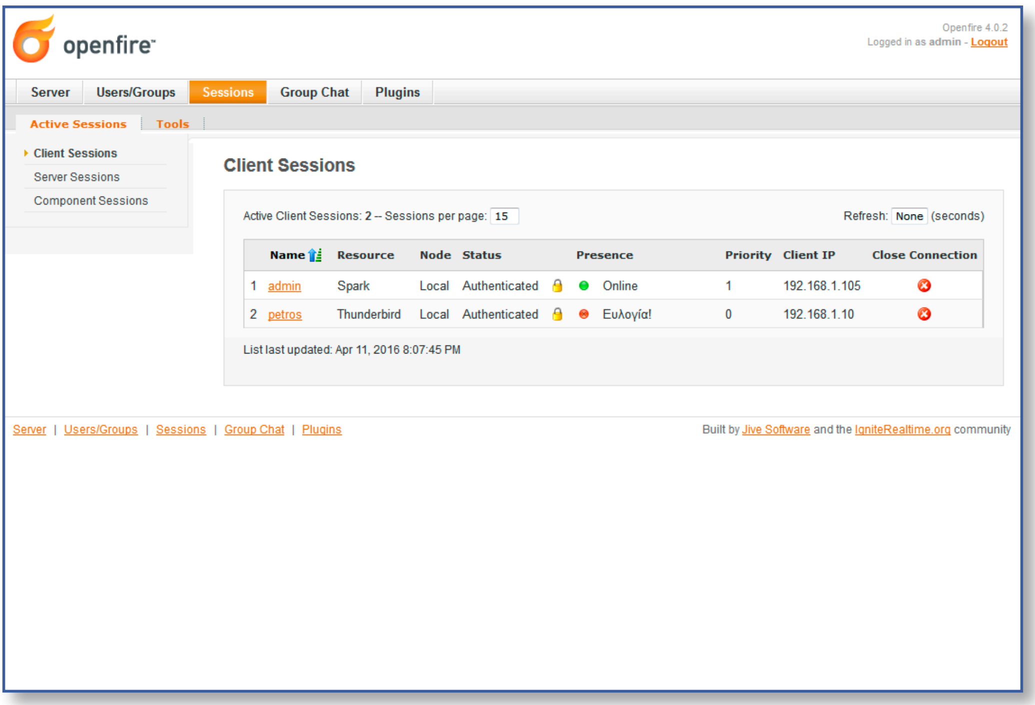


Τα βήματα ρύθμισης του OpenFire. Όπως βλέπετε επιλέξαμε την πιστοποίηση των χρηστών με τη συνδρομή της υπηρεσίας LDAP, που παρέχει ο domain controller του δικτύου μας.

Στο πρώτο βήμα επιλέγουμε γλώσσα. Αφήσαμε την προεπιλεγμένη (αγγλικά) και προχωρήσαμε. Μετά μπορούμε να δηλώσουμε το επιθυμητό όνομα για τον server και να ρυθμίσουμε κάποιες παραμέτρους του web interface. Εμείς επιλέξαμε το όνομα delta κι αφήσαμε όλα τα υπόλοιπα όπως ήταν. Σειρά είχε η επιλογή της βάσης δεδομένων που θα χρησιμοποιεί ο OpenFire. Επιτρέπεται να επιλέξουμε μια εξωτερική βάση που τρέχει στον υπολογιστή, όπως, για παράδειγμα, τη MySQL. Εναλλακτικά έχουμε το σύστημα HSQLDB, που ενσωματώνει ο OpenFire. Στις εγκαταστάσεις με πολλούς χρήστες προτείνεται η πρώτη επιλογή. Εμείς, πάντως, για να τσεκάρουμε την απόδοση του server επιλέξαμε την ενσωματωμένη βάση. Στο επόμενο βήμα καλούμαστε να επιλέξουμε το σύστημα διαχείρισης των χρηστών. Η προεπιλεγμένη ρύθμιση προβλέπει τη χρήση της βάσης δεδομένων. Προσφέρεται όμως και μια εναλλακτική επιλογή, που επιτρέπει τη διαχείριση των χρηστών μέσω LDAP. Όπως

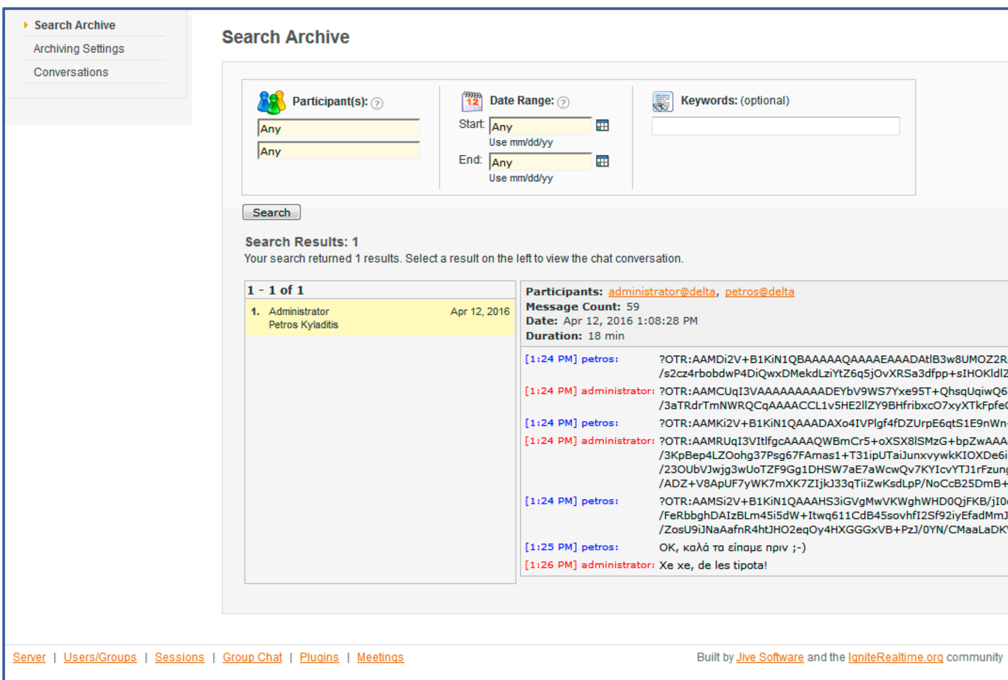
είπαμε νωρίτερα, σκοπεύουμε να αξιοποιήσουμε την υπηρεσία LDAP. Ωστόσο, στο πλαίσιο των δοκιμών αφήσαμε την εργοστασιακή ρύθμιση. Στο κάτω–κάτω, οι χρήστες που διαθέτουν την υπηρεσία LDAP στο τοπικό τους δίκτυο είναι ελάχιστοι και η χρήση του OpenFire χωρίς αυτήν είναι περισσότερο αντιπροσωπευτική. Στο επόμενο και τελευταίο βήμα, καλούμαστε να δώσουμε τα στοιχεία σύνδεσης του διαχειριστή. Η εγκατάσταση ολοκληρώνεται μόλις εισάγουμε τα επιθυμητά credentials.

Την επόμενη φορά που θα προσπαθήσουμε να μπούμε στο περιβάλλον διαχείρισης του OpenFire, θα χρειαστεί να δώσουμε τα στοιχεία σύνδεσης του διαχειριστή. Στην αρχική οθόνη του web interface του OpenFire βλέπουμε διάφορα στοιχεία και στατιστικά από τη λειτουργία του server. Μια από τις πρώτες εργασίες που θα χρειαστεί να κάνουμε είναι να προσθέσουμε χρήστες. Κάτι τέτοιο γίνεται πανεύκολα από την καρτέλα Users. Εκτός από μεμονωμένους χρήστες, μπορούμε να διαχειριζόμαστε και ολόκληρες ομάδες χρηστών. Τα groups, όπως ονομάζονται, διευκολύνουν την διαχείριση "δωματίων" ομαδικών συζητήσεων, που θυμίζουν τα chat room του IRC. Στα δωμάτια μπορούμε να επιτρέπουμε την ελεύθερη πρόσβαση σε όλους, στους χρήστες μιας ομάδας ή σε μεμονωμένους λογαριασμούς που διαθέτουν τον κωδικό του δωματίου. Η διαχείριση τέτοιων χώρων πραγματοποιείται από την καρτέλα Group Chat. Ενδιαφέρον συγκεντρώνει και η καρτέλα Sessions. Από εκεί μπορούμε να βλέπουμε την κατάσταση των συνδεδεμένων χρηστών και φυσικά να τους κάνουμε kick :P

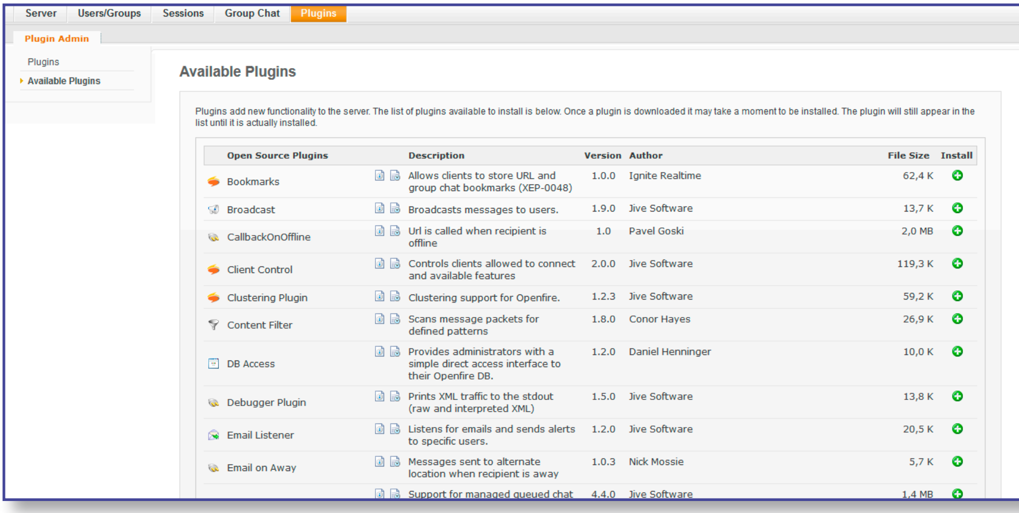


Οι ενεργές συνεδρίες των χρηστών του Jabber server. Με ένα απλό κλικ στο κόκκινο κουμπάκι με το "x" μπορούμε να πετάξουμε έξω όποιον χρήστη θέλουμε :D

Ξεχωριστή αναφορά αξίζει στην καρτέλα Plugins, μέσω της οποίας μπορούμε να προσθαφαιρούμε ...plugins. Εκεί προσφέρεται και μια μακροσκελής λίστα με πολύ ενδιαφέροντα πρόσθετα. Ανάμεσά τους, εμείς ξεχωρίσαμε κι εγκαταστήσαμε τα MotD (Message of the Day), Broadcast, Kraken IM Gateway και Monitoring Service. Με τη βοήθεια του πρώτου ορίζουμε ένα μήνυμα που αποστέλλεται αυτόματα σε κάθε χρήστη, αμέσως μετά τη σύνδεσή του. Το δεύτερο plugin διευκολύνει την άμεση αποστολή ενός μηνύματος στους ήδη συνδεδεμένους χρήστες (π.χ., "Μόλις έφτασαν οι πίτσες – τρέξτε!"). Το τρίτο πρόσθετο που εγκαταστήσαμε, μετατρέπει τον server μας σε gateway για πολλά διάσημα δίκτυα IM, όπως τα Facebook Messenger, Google Talk κ.ά. Τέλος, το Monitoring Service προσδίδει την ύπουλη δυνατότητα καταγραφής των συζητήσεων. Φυσικά, αυτό δεν το προσθέσαμε για κατασκοπευτικούς λόγους. Το χρησιμοποιήσαμε ελάχιστα, μόνο και μόνο για να τσεκάρουμε την αποτελεσματικότητα του OTR.



Η χρήση του OTR αχρηστεύει όλα τα κατασκοπευτικά μέσα που μπορεί να έχει επιστρατεύσει ένας αδιάκριτος διαχειριστής. Για του λόγου το αληθές πραγματοποιήσαμε μια δοκιμή με τη βοήθεια του Monitoring Plugin. Δείτε τι καταφέραμε να υποκλέψουμε.

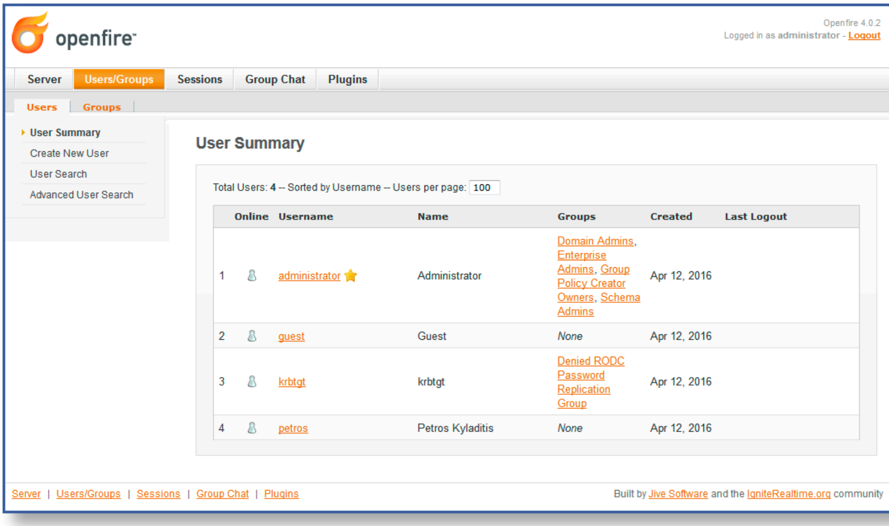


Ο OpenFire συνοδεύεται από μια πλούσια γκάμα plugins, που μπορούν να επεκτείνουν τη λειτουργικότητά του σε μεγάλο βαθμό. Η εγκατάσταση γίνεται πανεύκολα από τη σχετική καρτέλα.

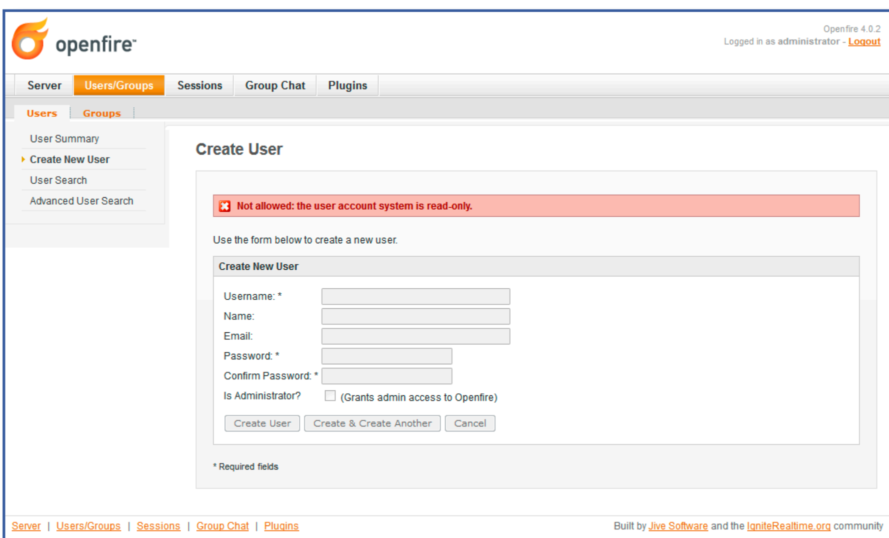
Διασύνδεση με τον domain controller

Όπως υποσχεθήκαμε στην αρχή, αντί για τη χειροκίνητη εισαγωγή χρηστών σκοπεύουμε να αξιοποιήσουμε τον domain controller του δικτύου μας. Για να πετύχουμε κάτι τέτοιο πρέπει να ρυθμίσουμε εξαρχής τον OpenFire κι αυτό γίνεται εύκολα με μια επανεγκατάσταση. Αυτή τη φορά, όταν βρεθήκαμε στον οδηγό ρυθμίσεων και στο βήμα Profile Setting, αλλάξαμε την επιλογή από Default σε Directory Server (LDAP). Έτσι προέκυψαν τρία νέα βήματα, που σχετίζονταν με τη ρύθμιση Directory Server. Στο πρώτο απ' αυτά (Connection Setting), στο πεδίο LDAP Server επιλέξαμε το Active Directory και στο πεδίο Host δώσαμε την IP του domain controller (στη δική μας περίπτωση ήταν 192.168.1.105). Για θύρα σύνδεσης ορίσαμε την 389. Κατόπιν, στο πεδίο Base DN πληκτρολογήσαμε το string "CN=Users,DC=delta,DC=local" (χωρίς τα εισαγωγικά). Κατά πάσα πιθανότητα, αυτά τα στοιχεία θα διαφέρουν στη δική σας εγκατάσταση. Ένα εργαλείο που θα σας βοηθήσει να τα βρείτε είναι το ADSI Edit, που αποτελεί έναν απλό LDAP Browser από την Microsoft. Στο πεδίο Authentication ορίσαμε τα στοιχεία πρόσβασης του διαχειριστή του domain controller και πραγματοποιήσαμε μια δοκιμή, κάνοντας κλικ στο Test Settings. Για να προχωρήσουμε στο επόμενο βήμα (User Mapping) πατήσαμε το κουμπί Save & Continue. Από εδώ μπορούμε να πραγματοποιήσουμε μια αντιστοιχία μεταξύ των δεδομένων που παρέχει ο LDAP για κάθε χρήστη και των δεδομένων που διατηρεί ο Jabber server. Ουσιαστικά, αντιστοιχούμε τα πεδία που παίρνουμε από τον ένα, στα πεδία που διατηρεί ο άλλος. Σε αυτό το βήμα, οι προεπιλεγμένες τιμές λειτουργούν μια χαρά και δεν χρειάστηκε να πειράξουμε κάτι. Το τρίτο βήμα (Group Mapping) προσφέρεται πάλι για την αντιστοίχιση δεδομένων, μόνο που αυτή τη φορά πρόκειται για στοιχεία ολόκληρων ομάδων και όχι μεμονωμένων χρηστών. Εδώ αφήσαμε και πάλι τις προεπιλεγμένες τιμές. Μετά από αυτά τα βήματα, ο οδηγός ρύθμισης του server ολοκληρώνεται και

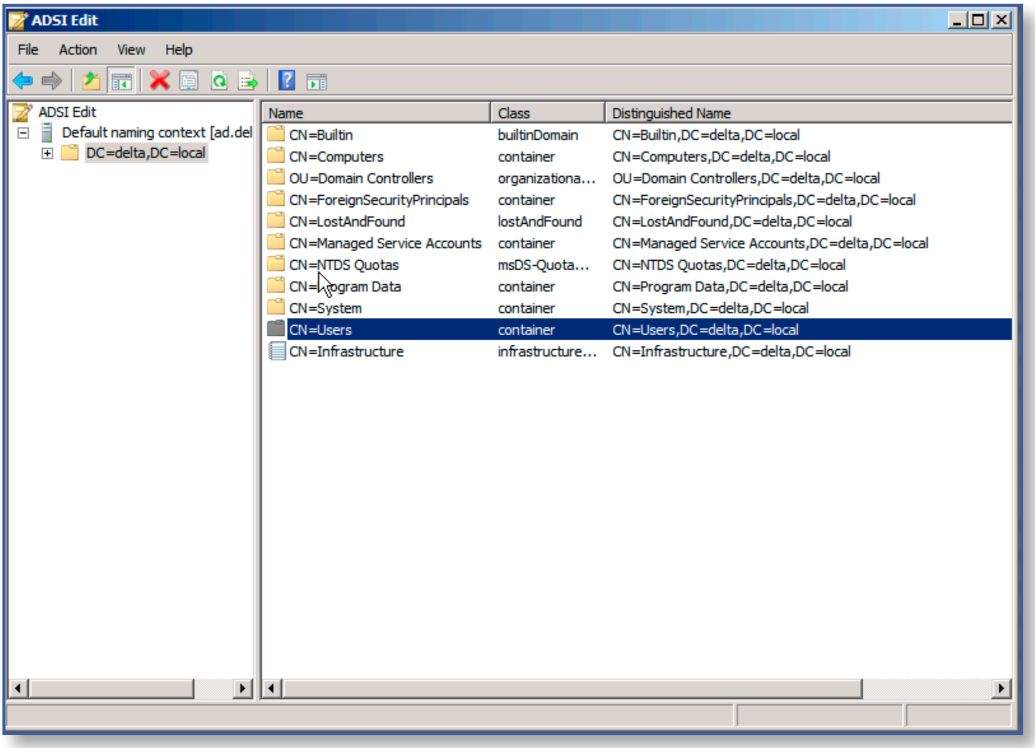
η ευθύνη για τη διαχείριση των χρηστών έχει μεταφερθεί στον domain controller. Σε αυτό το σημείο πρέπει να τονίσουμε μια σημαντική λεπτομέρεια: Πλέον, για να συνδεθούμε στο περιβάλλον διαχείρισης του OpenFire, πρέπει να δίνουμε τα στοιχεία σύνδεσης του διαχειριστή του domain! Επίσης, αν μεταβούμε στην καρτέλα Users θα δούμε όλους τους χρήστες που έχουμε δημιουργήσει για το domain και θα είναι αδύνατο να προσθέσουμε νέους. Αυτό δεν πρέπει να σας παραξενέψει. Ο domain controller έχει την αποκλειστική ευθύνη για τη διαχείριση των χρηστών.



Οι λογαριασμοί χρηστών του Jabber server, όπως ελήφθησαν από το active directory του domain controller.



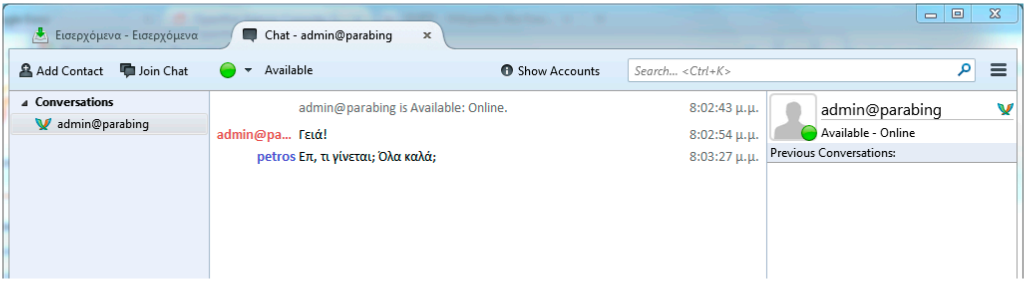
Όταν η ευθύνη διαχείρισης των χρηστών βρίσκεται στο domain controller, η δημιουργία νέων λογαριασμών από το web interface του OpenFire είναι απενεργοποιημένη.



Με τη βοήθεια του απλού LDAP Browser ονόματι ADSI Edit, μπορούμε να βρούμε με ευκολία το Base DN string, για τη σωστή ρύθμιση του OpenFire.

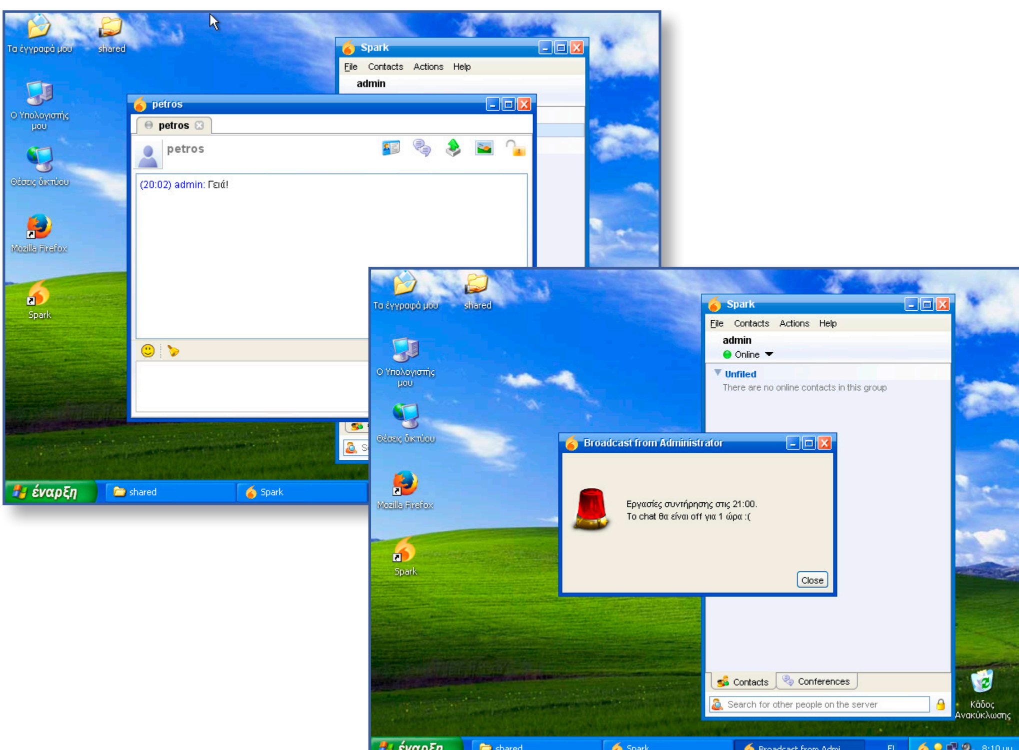
Από τον server στον client

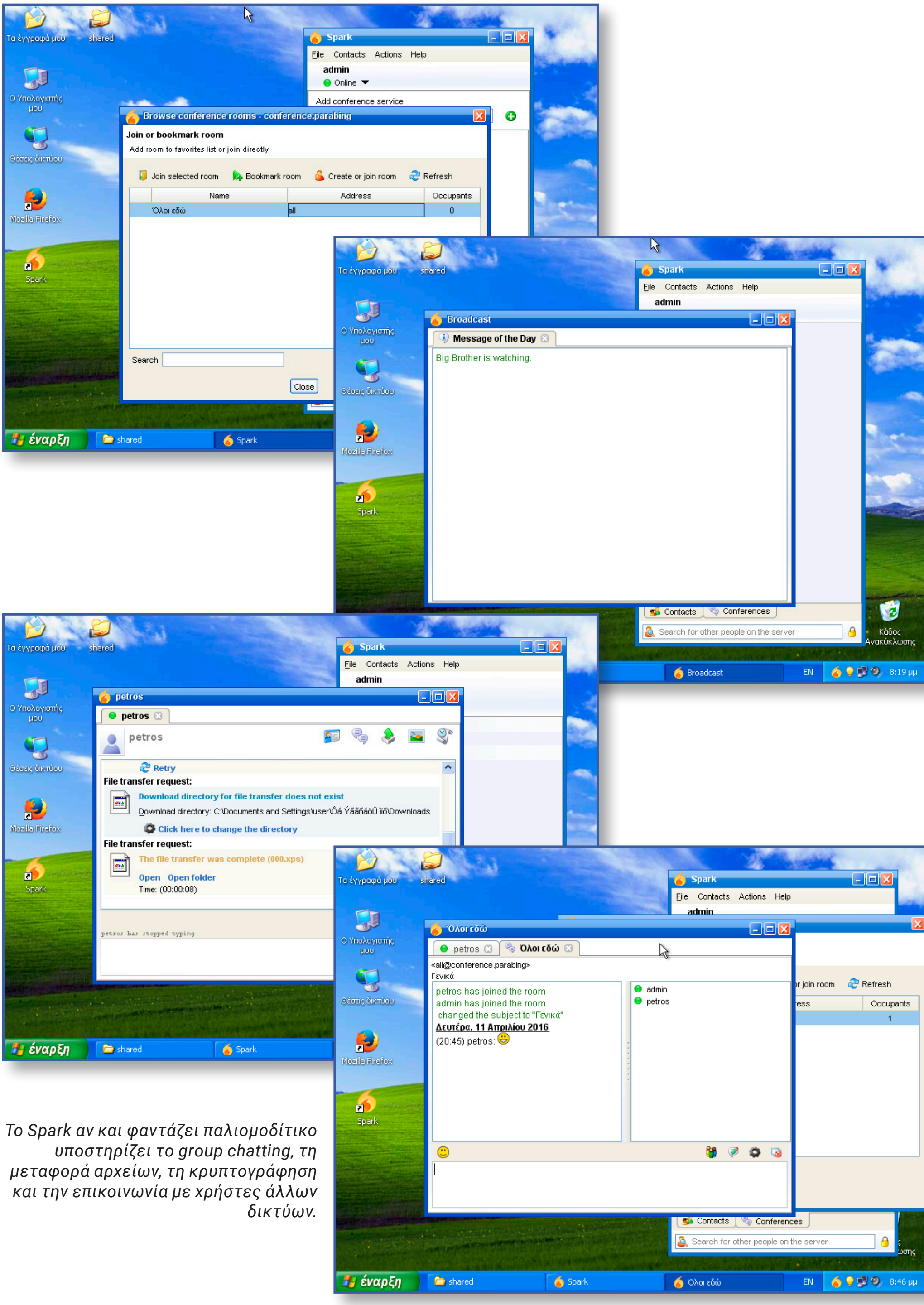
Όλα καλά με τις ρυθμίσεις του server, αλλά τι γίνεται με τους χρήστες; Οι ρυθμίσεις που είδαμε προηγουμένως ήταν αρκετά απλές, αλλά μέχρι στιγμής δεν έχουμε πει τίποτα για τους clients και όλο το εγχείρημα παραμένει, πρακτικά, θεωρητικό :P Για την ανταλλαγή μηνυμάτων μέσω του Jabber χρειαζόμαστε κάποιον IM Client που να υποστηρίζει το πρωτόκολλο. Τα προγράμματα που κυκλοφορούν είναι πάρα πολλά και η επιλογή τους επαφίεται στις ανάγκες αλλά και στα γούστα των χρηστών. Οποιαδήποτε εφαρμογή κι αν επιλέξουμε, εφόσον υποστηρίζει το πρωτόκολλο Jabber θα μας επιτρέψει να επικοινωνήσουμε με όλους τους υπόλοιπους χρήστες του server, ανεξάρτητα από τον client που έχουν επιλέξει. Εμείς δοκιμάσαμε αρχικά την αποστολή μηνυμάτων μέσω Thunderbird (ναι, υποστηρίζει *και* Jabber). Από το παράθυρο διαχείρισης λογαριασμών επιλέξαμε τη δημιουργία ενός νέου. Ως τύπο επιλέξαμε το XMPP. Ακολούθως δώσαμε τα στοιχεία σύνδεσης που έχουμε ως απλός χρήστης του domain και στο πεδίο server πληκτρολογήσαμε την IP του μηχανήματος που τρέχει τον OpenFire. Η σύνδεση ήταν επιτυχής, ενώ μας δόθηκε η ευκαιρία να προσθέσουμε και μια νέα επαφή, από τους χρήστες που είχε ο server. Η ανταλλαγή μηνυμάτων γινόταν χωρίς προβλήματα, αλλά οι δυνατότητες του client που ενσωματώνει ο Thunderbird είναι περιορισμένες: ούτε η ανταλλαγή αρχείων επιτρέπεται, ούτε OTR προσφέρεται.



Ο Thunderbird υποστηρίζει την ανταλλαγή μηνυμάτων μέσω Jabber, αλλά μέχρι εκεί. Ούτε αρχεία μεταφέρει, ούτε κρυπτογράφηση προσφέρει.

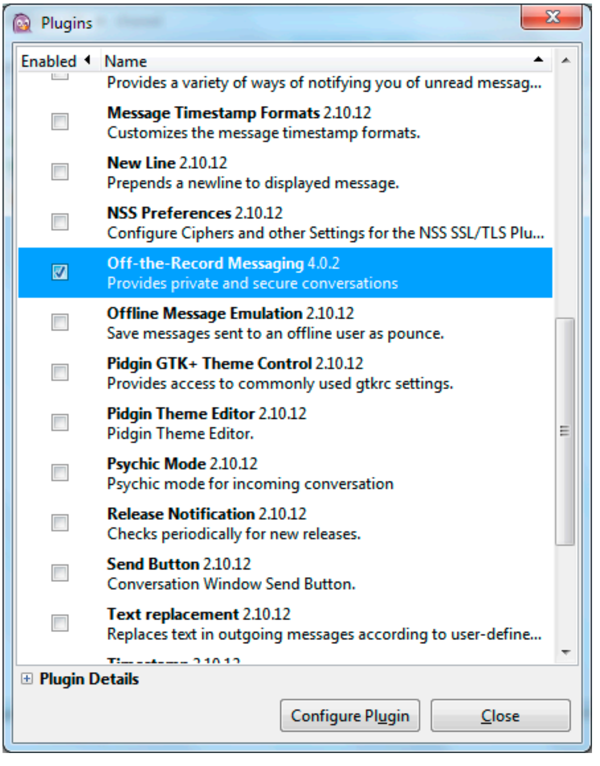
Ο επόμενος client που δοκιμάσαμε ήταν ο Spark. Είναι γραμμένος σε Java και προέρχεται από την ίδια κοινότητα που ανέπτυξε τον OpenFire. Η εγκατάσταση και η σύνδεση με το server έγιναν γρήγορα και εύκολα. Ο Spark έκανε τα πάντα! Κατ' αρχάς, μπορούσαμε να συνδεθούμε σε άλλα δίκτυα IM, όπως το Google Talk, χρησιμοποιώντας ως ενδιάμεσο κρίκο το Jabber Server. Επιπρόσθετα, το Spark επιτρέπει τη μεταφορά αρχείων, τη σύνδεση σε chat room και την κρυπτογράφηση των μηνυμάτων μας. Ένας τρίτος client που δοκιμάσαμε είναι ο αγαπημένος μας Pidgin. Περιττό να πούμε ότι υποστηρίζει πλήρως το XMPP και την ανταλλαγή αρχείων, ενώ με τη βοήθεια ενός plugin που μπορούμε να βρούμε στη διεύθυνση https://otr.cypherpunks.ca υποστηρίζει και την προστασία των μηνυμάτων μας μέσω OTR.



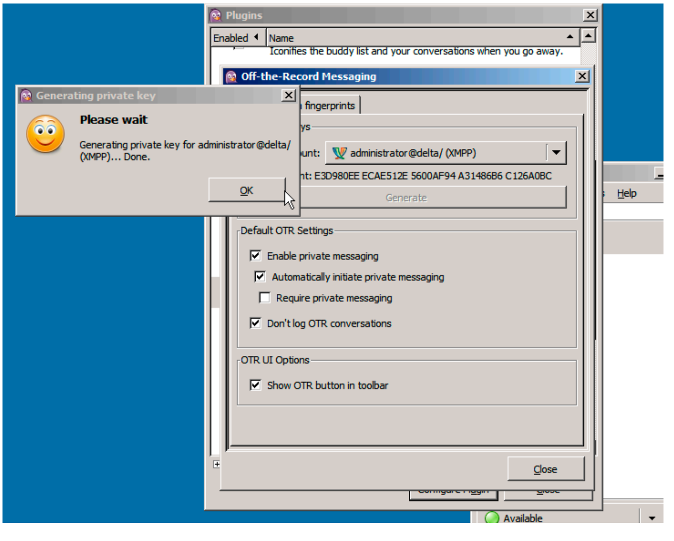


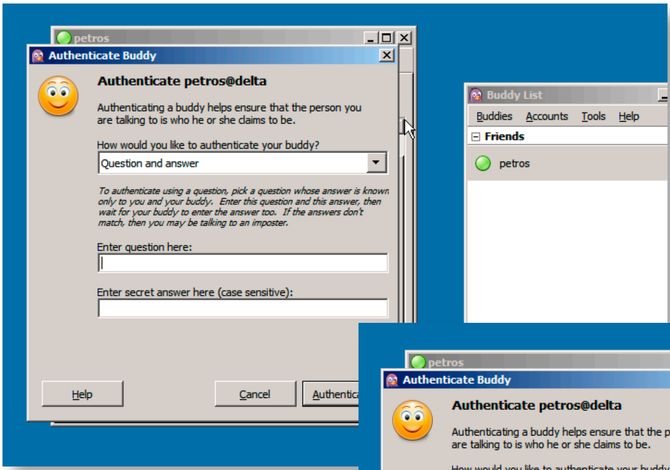
Το Spark αν και φαντάζει παλιομοδίτικο υποστηρίζει το group chatting, τη μεταφορά αρχείων, τη κρυπτογράφηση και την επικοινωνία με χρήστες άλλων δικτύων.

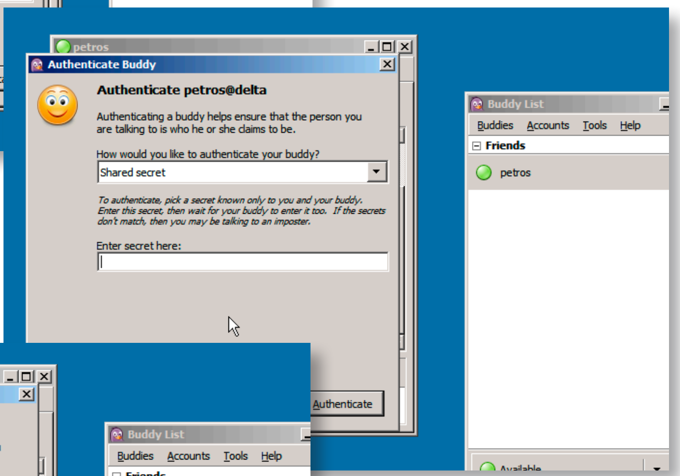
Το XMPP αποτελεί πρωτόκολλο-ευλογία! Κάνει τα πάντα, είναι ανοικτό, αποκεντρωμένο και συνεργάζεται με όλα τα δημοφιλή δίκτυα IM. Εσείς μπορείτε ν' αντισταθείτε σ' έναν τέτοιο πειρασμό;

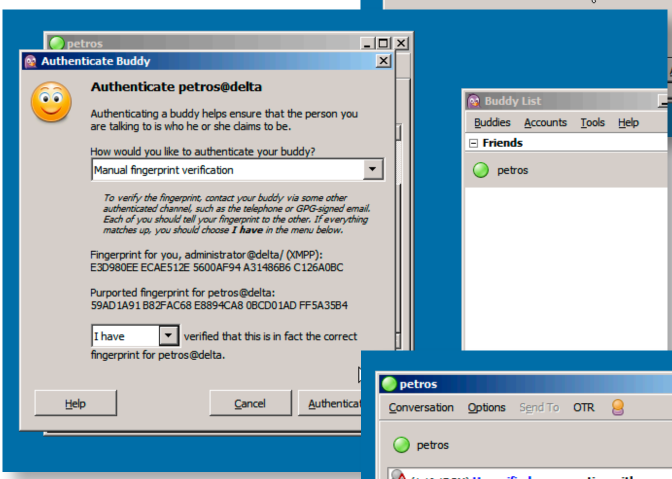


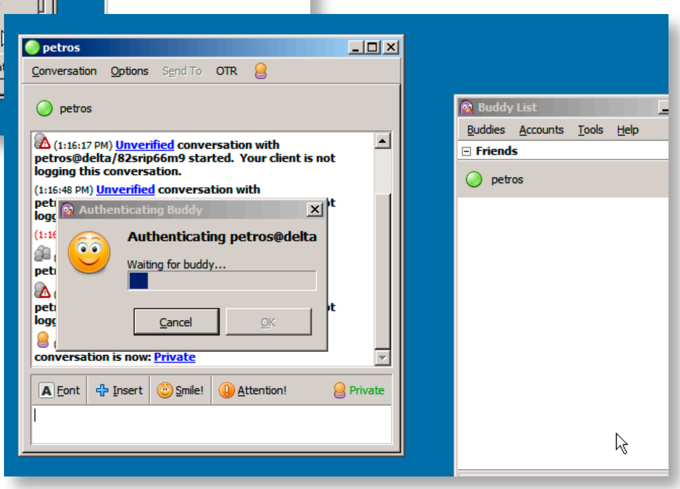
Η κρυπτογράφηση end-to-end με τη μεσολάβηση του OTR υποστηρίζεται και στο pidgin. Αρκεί να ενεργοποιήσουμε το σχετικό plugin.











Το OTR plug-in για το Pidgin φροντίζει τα πάντα. Χρησιμοποιεί τεχνικές ασύμμετρης κρυπτογράφησης και προσφέρει πολλαπλές μεθόδους για την πιστοποίηση του δημόσιου κλειδιού του χρήστη. Αφού το ρυθμίσουμε, μπορούμε να επικοινωνήσουμε άφοβα με οποιονδήποτε.

Skill: Intermediate
Tags: Unity, games, game loop, Bezier curves

Ο κεντρικός ρόλος του game loop

Στο άρθρο που ξεκινά από τη σελίδα 22 του τεύχους, αρχίσαμε να στήνουμε το παιχνίδι μας κάνοντας τη σφαίρα που θα ελέγχει ο παίκτης να κινείται αριστερά–δεξιά. Αυτή τη φορά θα μας απασχολήσει λίγη θεωρία. Ναι, και σε μας δεν αρέσει η θεωρία, αλλά ξεκινώντας από αυτή θα μπορέσουμε να κατανοήσουμε ακόμα περισσότερο πώς λειτουργεί ο κώδικας των παιχνιδιών.

του Γιάννη Κρομμύδα

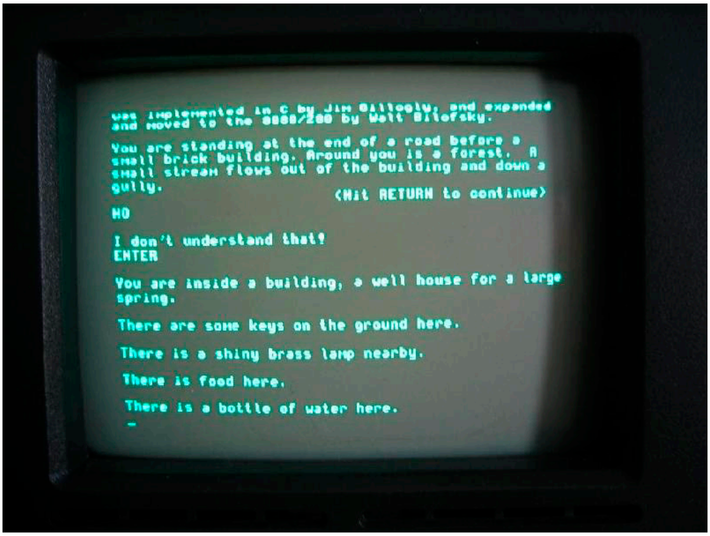
Όπως αναφέραμε επιγραμματικά στο προηγούμενο άρθρο, το λεγόμενο game loop είναι ένα programming pattern και χρησιμοποιείται για να αποσυνδέουμε την εξέλιξη ενός παιχνιδιού από την είσοδο του χρήστη. Προκειμένου να δούμε πώς μας προέκυψαν αυτά τα game loops, χρειάζεται να πάμε κάποια χρόνια πίσω.

A long, long time ago...

Τον παλιό καιρό, τότε που οι οθόνες ήταν ασπρόμαυρες, κιτρινόμαυρες ή πρασινόμαυρες, ανάλογα με τη μάρκα, τα video games είχαν διαφορετική δομή σε σχέση με τα σύγχρονα. Δεν θα ήταν λάθος αν λέγαμε πως η δομή τους ήταν άμεση συνέπεια του επιπέδου της τότε τεχνολογίας. Ας μην ξεχνάμε άλλωστε –οι μικρότεροι ας μαθαίνουν– ότι η υπολογιστική ισχύς των επεξεργαστών την περίοδο εκείνη ήταν στα επίπεδα που είναι σήμερα τα πλυντήρια και τα ψυγεία. Προγραμματίζαμε τους τότε υπολογιστές, πατούσαμε ένα κουμπί και περιμέναμε να πάρουμε τ' αποτελέσματα. Ίσως θα παρατηρήσατε ότι από τη συγκεκριμένη διαδικασία λείπει κάτι: Πέρα από το πάτημα του κουμπιού για την έναρξη των υπολογισμών, ο παράγοντας της ανθρώπινης αλληλεπίδρασης είναι ανύπαρκτος. Ουσιαστικά, μόλις αναφερθήκαμε στην επεξεργασία σύμφωνα με το μοντέλο του batch processing. Τα προγράμματα που ακολουθούν την εν λόγω λογική λέμε ότι λειτουργούν σε batch mode. Θα έχετε προσέξει τα αρχεία .bat στα Windows, τα λεγόμενα batch files – έτσι δεν είναι;

Κατά βάση, τα προγράμματα που συμμορφώνονται στη λογική του batch processing επιστρατεύονται, ακόμη και σήμερα, για εργασίες που εκτελούνται χωρίς την ανάγκη αλληλεπίδρασης με τον χειριστή (δηλαδή με τον χρήστη). Τα δεδομένα εισόδου που δέχονται είναι πιθανό να προέρχονται, π.χ., από άλλα batch files ή scripts ή/και από παραμέτρους που περνάμε αρχικά, στο command line. Είναι λοιπόν προφανές ότι τέτοια προγράμματα είναι κατάλληλα για τον αυτοματισμό μηχανικών/συνηθισμένων διαδικασιών.

Καθώς περνούσε ο καιρός άρχισε να γίνεται φανερή η ανάγκη και για προγράμματα ικανά για αλληλεπίδραση με τον χρήστη: όχι μόνο αρχικά, κατά την ενεργοποίησή τους, αλλά και κατά το χρόνο εκτέλεσης. Σιγά σιγά λοιπόν άρχισαν να γράφονται τα πρώτα διαδραστικά (interactive) προγράμματα. Μερικά από τα πρώτα προγράμματα του είδους ήταν τα παιχνίδια. Οι παλιότεροι πιθανώς θα θυμόσαστε τα πρώτα text adventure games. Σε αυτά, βλέπαμε στην οθόνη μας το κείμενο που περιέγραφε το περιβάλλον, δηλαδή το τι μπορούσε να βλέπει ο χαρακτήρας του παίκτη, κι ο υπολογιστής ανέμενε εντολές κειμένου από τον παίκτη ώστε να προχωρήσει αναλόγως.



Το Colossal Cave Adventure ή ADVENT τρέχει σε έναν υπολογιστή Osborne 1, γύρω στο 1982. Το παιχνίδι γράφτηκε το '76 από τον Will Crowther. Ο παίκτης καλείται να εξερευνήσει μια σπηλιά, στην οποία φήμες λέγαν ότι υπήρχε ένας αμύθητος θησαυρός. Η αλληλεπίδραση με τον παίκτη γινόταν με απλές εντολές κειμένου.

Θα μπορούσαμε να πούμε ότι γινόταν ένας διάλογος μεταξύ του παίκτη και του υπολογιστή. Ο υπολογιστής περίμενε την είσοδο του παίκτη και αντιδρούσε αναλόγως.

Επιστροφή στο σήμερα

Με παρόμοιο τρόπο δουλεύουν και οι μοντέρνες εφαρμογές για το περιβάλλον γραφικών. Η μόνη διαφορά με τα αρχαία προγράμματα είναι ότι αντί για μια εντολή κειμένου, το πρόγραμμα περιμένει ένα event. Ο επεξεργαστής απλά κάθεται μέχρι ο χρήστης να πατήσει ένα κουμπί ή ένα πλήκτρο. Μπλοκάρεται, δηλαδή, η εκτέλεση του προγράμματος, μέχρι να υπάρξει είσοδος από τον χρήστη.

```
while (!done)
{
      Event event = waitForEvent();
      dispatchEvent(event);
}
```

Όμως τα σημερινά παιχνίδια βλέπουμε ότι συνεχίζουν να εκτελούνται (οι NPCs κινούνται, τα particles συνεχίζουν να χορεύουν αριστερά–δεξιά) ακόμη κι όταν ο παίκτης κάθεται κι απλά κοιτάζει την οθόνη. Το παιχνίδι δεν παγώνει, περιμένοντας τον παίκτη να ανταποκριθεί. Ακριβώς αυτό, φίλες και φίλοι, είναι το πρώτο, κύριο σημείο αναφοράς ενός game loop: Επεξεργάσου την είσοδο του παίκτη, αλλά *μην* την περιμένεις. Σύμφωνα μ' αυτή τη λογική, το loop συνεχίζει κι εκτελείται:

```
while (!done)
{
      processInput();
      updateGameLogic();
      renderToScreen();
}
```

Αφού πλέον το loop εκτελείται διαρκώς, φτάνουμε στα εξής ερωτήματα: Πόσο γρήγορα εκτελείται το game loop; Σε κάθε εκτέλεσή του, το παιχνίδι προχωράει λίγο την εξέλιξή του. Πόσο όμως πρέπει να εξελίσσουμε το παιχνίδι; Πρέπει, π.χ., να καταστραφεί το κτίριο που είχε πάρει φωτιά μετά την επίθεση του εχθρού ή όχι ακόμα;

Αν μετρήσουμε το πόσες φορές εκτελείται το loop ανά δευτερόλεπτο, παίρνουμε τον αριθμό των FPS (frames per second) του παιχνιδιού. Παλιότερα, μιας και ο αριθμός των διαθέσιμων μηχανημάτων ήταν πεπερασμένος, οι προγραμματιστές ήξεραν τα χαρακτηριστικά κάθε μηχανήματος κι έτσι μπορούσαν να υπολογίζουν τον πραγματικό χρόνο (σε ms) μεταξύ δύο διαδοχικών επαναλήψεων του loop. Το αποτέλεσμα ήταν ένα ομαλό gameplay.

Κατά τα αναμενόμενα, τα πράγματα σήμερα είναι αρκετά πιο δύσκολα. Σε γενικές γραμμές, καλύτερο μηχάνημα σημαίνει και μεγαλύτερος αριθμός FPS. Καλύτερο μηχάνημα όμως σημαίνει και ακριβότερο μηχάνημα, κι αυτή είναι μια πολυτέλεια που δεν μπορεί να έχει ο καθένας. Τις περισσότερες φορές άλλωστε, δεν γίνεται να γνωρίζουμε σε τι μηχάνημα θα τρέξει το παιχνίδι που θα φτιάξουμε. Υπάρχει τόσο μεγάλη ποικιλία εξαρτημάτων/υποσυστημάτων στο εμπόριο, που οι δυνατοί συνδυασμοί –αν όχι άπειροι– είναι απαγορευτικά πολλοί ώστε να τους λάβουμε όλους υπόψη.

Προκειμένου να λύσουμε *και* αυτό το πρόβλημα, οφείλουμε να προγραμματίσουμε το παιχνίδι προκειμένου να προσαρμόζεται στο εκάστοτε hardware. Κι αυτό ακριβώς είναι το δεύτερο κύριο σημείο αναφοράς ενός game loop: τρέξε το παιχνίδι με την ίδια ταχύτητα *ανεξάρτητα* από το μηχάνημα που εκτελείσαι.

The Game Loop

Φτάνουμε τώρα στην έννοια του pattern. Κατά τη διάρκεια του gameplay, το game loop πρέπει να τρέχει χωρίς να σταματά. Οφείλει να επεξεργάζεται την είσοδο του χρήστη χωρίς να μπλοκάρει, να εξελίσσει την κατάσταση του παιχνιδιού και να

προβάλλει την τρέχουσα κατάσταση στον παίκτη. Τέλος, πρέπει να ανιχνεύει την πάροδο του χρόνου και να διαχειρίζεται το ρυθμό της εξέλιξης του gameplay. Δεν ξέρουμε πώς σας ακούγονται αυτές οι υποχρεώσεις, δεν είναι όμως παίξε γέλασε. (Κάπως ειρωνικό αυτό, αν σκεφτούμε ότι σκοπός μας εδώ είναι να παίζουμε – γιατί όχι και να γελάμε.)

Στη δική μας περίπτωση, η αλήθεια είναι ότι το game loop είναι βαθιά ριζωμένο στα ενδότερα της Unity. Δεν θα χρειαστεί, λοιπόν, να το προγραμματίσουμε. Τις γνώσεις περί game loop όμως, έχουμε την άριστη ευκαιρία να τις χρησιμοποιήσουμε ώστε να καταλάβουμε καλύτερα πώς δουλεύει η μηχανή.

Πίσω στο παιχνίδι μας

Ας επιστρέψουμε στον κώδικα του προηγούμενου άρθρου κι ας τον ξαναδούμε υπό το φως όλων όσων μάθαμε.

```
using UnityEngine;
namespace Deltahacker
{

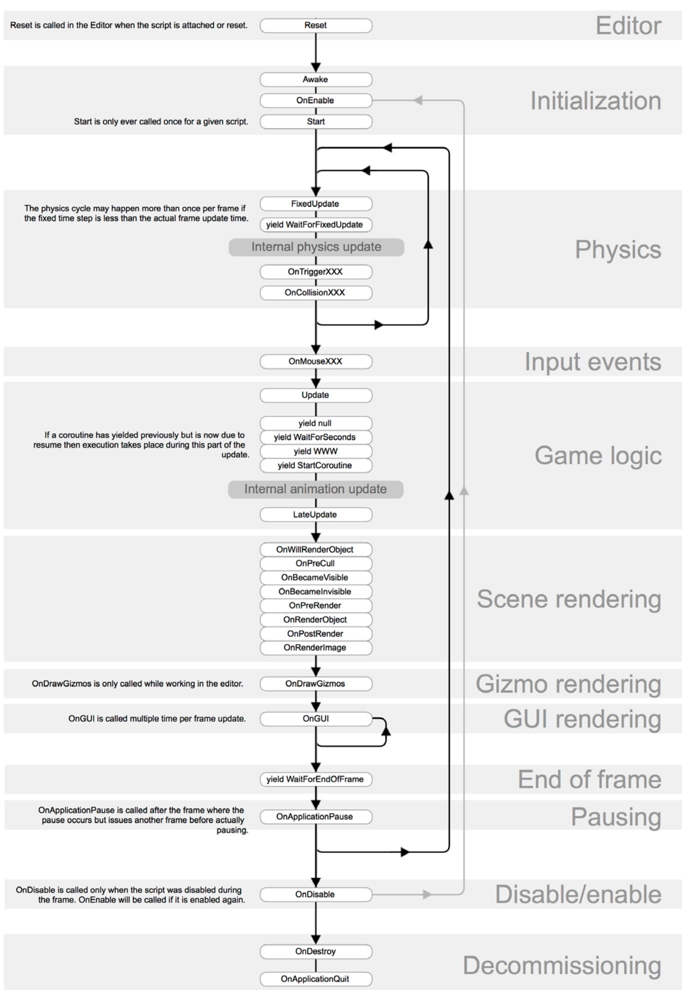
      public class Player : MonoBehaviour
      {
         public float Speed = 3f;

         public void Update()
         {
            float horizontal = Input.GetAxis("Horizontal");
            Vector3 movement = new Vector3(horizontal * Speed * Time.deltaTime,
            0, 0);
            transform.position += movement;
         }
      }
}
```

Υλοποιήσαμε τη μέθοδο Update, της κλάσης Player. Για τον υπολογισμό του διανύσματος της κίνησης της σφαίρας χρησιμοποιήσαμε την κλάση Time, της Unity. Η ενσωματωμένη αυτή κλάση είναι άμεσα συνδεδεμένη με το game loop της μηχανής κι ανά πάσα στιγμή μας δίνει πληροφορίες για το χρόνο του παιχνιδιού. Συγκεκριμένα, η static μεταβλητή deltaTime επιστρέφει το χρόνο που έχει περάσει από την προηγούμενη εκτέλεση του loop. Δηλαδή μας λέει πόσος χρόνος πέρασε από την προηγούμενη φορά που εκτελέστηκε η μέθοδος Update. Η τιμή της είναι συνήθως της τάξης των milliseconds, αλλά επιστρέφεται πάντα ως seconds. Παραδείγματος χάριν, στα 60 fps η τιμή της θα είναι 1s / 60 fps = 0,01667 seconds. Ο πολλαπλασια-

σμός της ταχύτητας της σφαίρας με την τιμή αυτή μας βοηθάει να κινούμε με σταθερή ταχύτητα τη σφαίρα, χωρίς να μας ενδιαφέρει η ταχύτητα στην οποία τρέχει το παιχνίδι.

Θα θέλαμε να σχολιάσουμε επίσης την κλάση MonoBehaviour, από την οποία κληρονομεί η κλάση που γράψαμε. Για να προσθέσουμε ένα script σε ένα GameObject που βρίσκεται στην σκηνή μας (είτε όχι, αν είναι Prefab), το script πρέπει να είναι απόγονος της κλάσης MonoBehaviour. Η κλάση είναι εσωτερική της Unity και περιλαμβάνει τις μεθόδους και τα events που χρειάζονται για την αλληλεπίδραση ή το χειρισμό των αντικειμένων στο παιχνίδι μας. Αργότερα θα υλοποιούμε περισσότερες από αυτές τις μεθόδους σε κάθε script μας, οπότε καλό θα ήταν να γνωρίζουμε τη σειρά με την οποία αυτές εκτελούνται.



Το flowchart της Unity, όπως δίνεται στο επίσημο manual. Η μέθοδος Update που είδαμε στο script του προηγούμενου άρθρου, αποτελεί μέρος του Game Logic.

Δεν είναι απαραίτητο όλες οι κλάσεις που γράφουμε να είναι απόγονοι της κλάσης MonoBehaviour. Ίσα ίσα που καλό θα ήταν να χρησιμοποιούμε την κλάση μόνο σε όσα scripts χρειάζεται. Το πού χρειάζεται και πού όχι, μπορούμε να το καταλάβουμε αναλογιζόμενοι αν το αντικείμενο χρειάζεται να κάνει υπολογισμούς σε κάθε frame από μόνο του. Άλλες φορές, η εμπειρία μας θα μας λέει ότι η εκάστοτε κλάση δεν χρειάζεται να είναι απόγονος της MonoBehaviour.

When I say jump…

Μετά τον ορισμό του game loop και τη δεύτερη ματιά στον κώδικα της κίνησης, ήρθε η ώρα να κάνουμε τη σφαίρα που κινεί ο παίκτης να πηδάει από γραμμή σε γραμμή. Ανοίγουμε το project στη Unity και πάμε να τροποποιήσουμε το υπάρχον script μας.

Υπάρχουν αρκετοί τρόποι να κάνουμε την σφαίρα να κινείται. Άλλοι περιλαμβάνουν γραφίστες (ενσωματωμένο animation στο μοντέλο της σφαίρας), ενώ άλλοι περιλαμβάνουν τον ορισμό ενός animation με τη βοήθεια της ενσωματωμένης μηχανής που προσφέρει η Unity. Θα καταφύγουμε σε έναν πιο απλό τρόπο, παίζοντας κατευθείαν με τη θέση της σφαίρας. Ας προσθέσουμε λοιπόν τον κώδικα που χρειάζεται για το άλμα στην κλάση Player, κι ας τον μελετήσουμε λεπτομερώς.

```
using UnityEngine;
namespace Deltahacker
{
public class Player : MonoBehaviour
{
          public float Speed = 3f;
          [Header("Jump Setup")]
          public float JumpHeight = 1f;
          public float JumpDistance = 2f;
          public float JumpDuration = 0.5f;
          public AnimationCurve JumpCurve;

          private bool jumping;
          private float jumpDirection;
          private float currentJumpTime;
          private float normalY;
          private float relativeCurveTime;
          public void Start()
          {
             normalY = transform.position.y;
```

```
        float curveTime = JumpCurve[JumpCurve.length - 1].time;
        relativeCurveTime = curveTime / JumpDuration;
    }
    public void Update()
    {
        float horizontalAxis = Input.GetAxis("Horizontal");
        move(horizontalAxis);
        float verticalAxis = Input.GetAxis("Vertical");
        if (jumping)
            handleJump();
        else if (verticalAxis != 0)
            jump(verticalAxis);
    }
    private void move(float axis)
    {
        Vector3 movement = new Vector3(axis * Speed * Time.deltaTime, 0, 0);
        transform.position += movement;
    }
    private void jump(float axis)
    {
        if (axis == 0)
            return;
        jumpDirection = axis < 0 ? -1 : 1;
        if (jumpDirection != 0)
        {
            currentJumpTime = 0;
            jumping = true;
            handleJump();
        }
    }
    private void handleJump()
    {
        currentJumpTime += Time.deltaTime;
        float animationEval = JumpCurve.Evaluate(currentJumpTime *
        relativeCurveTime);
```

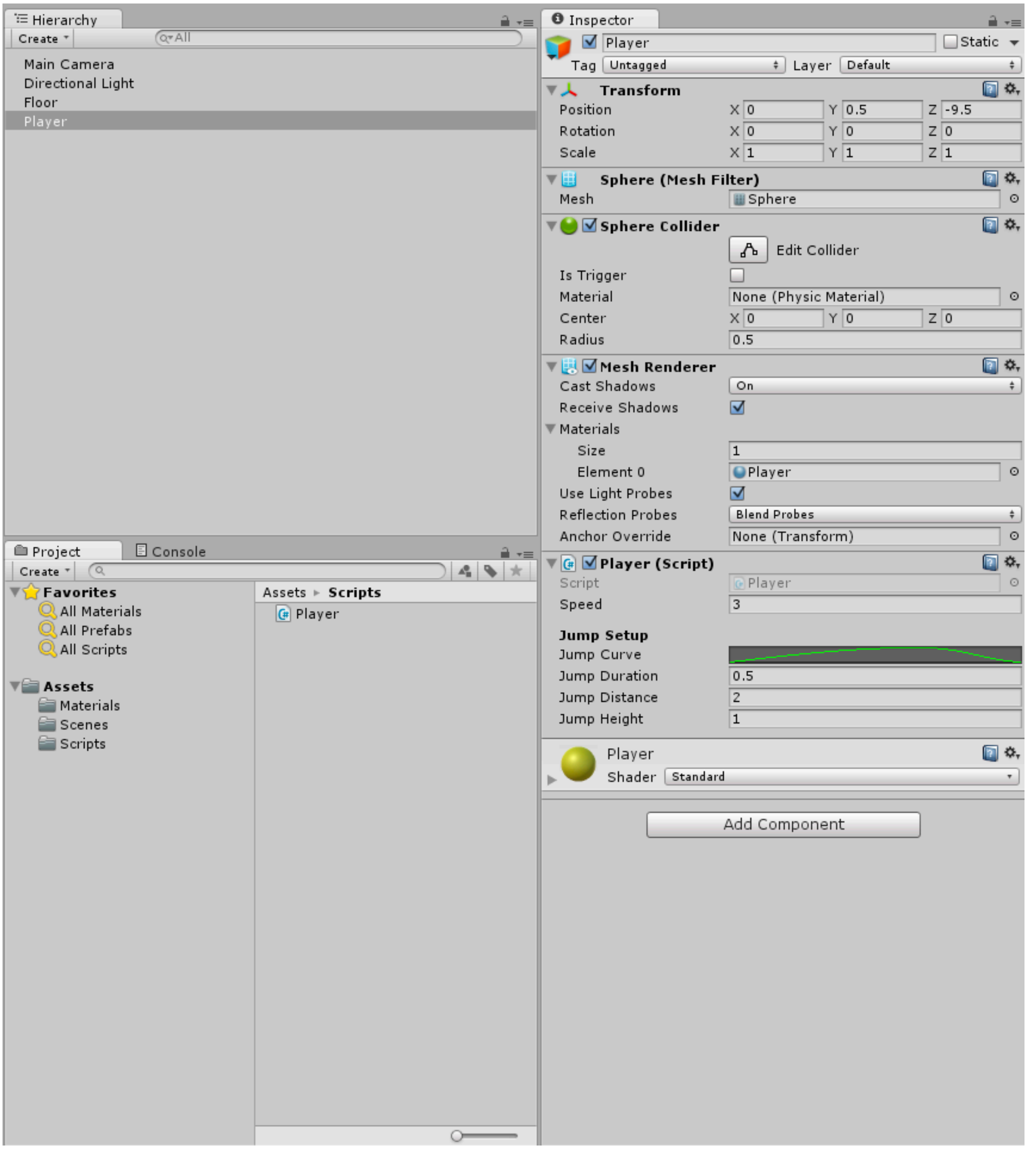
```
            float currentHeight = normalY + (animationEval * JumpHeight);
            float forwardMovement = jumpDirection * JumpDistance * Time.
            deltaTime;
            Vector3 jumpVector = new Vector3(transform.position.x,
            currentHeight, transform.position.z + forwardMovement);
            transform.position = jumpVector;
            if (currentJumpTime >= JumpDuration)
               jumping = false;
        }
    }
}
```

Αρχικά κάναμε refactoring στην κλάση. Μετακινήσαμε τον κώδικα που είναι υπεύθυνος για την κίνηση της σφαίρας αριστερά–δεξιά στη δικιά του private μέθοδο, με όνομα move. Αντίστοιχα με τη move δημιουργήσαμε μια νέα μέθοδο jump, με μοναδικό όρισμα ένα float, η οποία θα μας υποδεικνύει προς ποια κατεύθυνση θέλει ο παίκτης να κάνει άλμα η σφαίρα. Η μέθοδος αυτή θα είναι υπεύθυνη μόνο για την έναρξη του άλματος, οπότε θα χρειαστούμε άλλη μία μέθοδο μέσα στην οποία θα τροποποιούμε τη θέση της σφαίρας, ώστε να υλοποιήσουμε το άλμα. Η μέθοδος αυτή ονομάζεται handleJump. Παράλληλα, ορίσαμε μια σειρά από public και private μεταβλητές για να μας βοηθήσουν στη διαδικασία του άλματος.

Αλλάζοντας τις τιμές των public πεδίων JumpHeight, JumpDistance, JumpDuration και JumpCurve, θα μπορούμε να μεταβάλλουμε τη συμπεριφορά του άλματος χωρίς να χρειάζεται να επεμβαίνουμε στον κώδικά μας. Αναλυτικότερα, το JumpHeight μεταβάλλει το ύψος του άλματος, το JumpDistance μεταβάλλει το μήκος του, ενώ το JumpDuration μεταβάλλει τον χρόνο που χρειάζεται για να ολοκληρωθεί το άλμα. Ο τύπος δεδομένων των τριών αυτών μεταβλητών είναι float. Η τέταρτη μεταβλητή θα ορίζει την καμπύλη του άλματος, για αυτό και χρησιμοποιούμε την ενσωματωμένη κλάση AnimationCurve. Περισσότερα γι' αυτή, λίαν συντόμως. Χρησιμοποιώντας το attribute Header μπορούμε να προσθέσουμε μια κεφαλίδα στις τιμές που φαίνονται στον inspector, ομαδοποιώντας (οπτικά μόνο) τις τιμές που εμφανίζονται.

Τα private πεδία είναι περισσότερο βοηθητικά. Η boolean μεταβλητή jumping θα καθίσταται true μόνον όταν η σφαίρα έχει ξεκινήσει το άλμα της. Στη jumpDirection θα αποθηκεύουμε την τιμή του κάθετου άξονα (πλήκτρα [W] και [S]) όταν ξεκινάει το άλμα. Με αυτόν τον τρόπο, αν ο χρήστης κατά τη διάρκεια του άλματος αλλάξει την τιμή του άξονα δεν θα επηρεαστεί η κίνηση της σφαίρας. Η currentJumpTime θα κρατάει την ώρα που πέρασε από την αρχή του άλματος (μια και θέλουμε το άλμα να σταματήσει με το πέρας του JumpDuration). Στη μεταβλητή normalY θα αποθηκεύσουμε την αρχική θέση της σφαίρας, στον κατακόρυφο άξονα y. Η relativeCurveTime θα μας βοηθήσει να αναγάγουμε το χρόνο που είναι δηλωμένος στην καμπύλη του άλματος, στο χρόνο που χρειάζεται για να ολοκληρωθεί το άλμα.

Ας μιλήσουμε λίγο για την καμπύλη του άλματος. Η κλάση AnimationCurve είναι μια συλλογή από καμπύλες (Bezier curves) που υποδηλώνουν τη συνάρτηση που τροποποιεί μια τιμή. Επιστρέφοντας στη Unity (αφού σώσουμε τον κώδικά μας) και βλέποντας τον Inspector του αντικειμένου Player, θα παρατηρήσουμε ότι στο component Player που ορίζεται από το script μας έχουν προστεθεί τα public πεδία που έχουμε ορίσει στην κλάση.

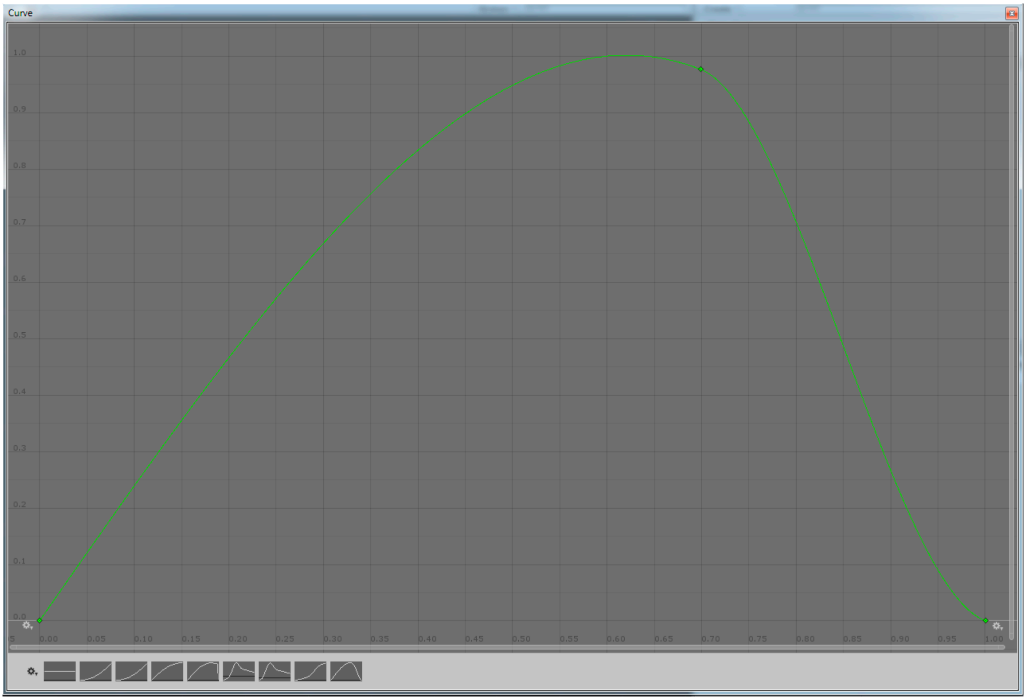


Αλλάζοντας στον κώδικα τα public πεδία των κλάσεων που έχουμε προσκολλήσει σε ένα game object, θα δούμε ότι οι αλλαγές αυτές αντανακλώνται και στο παράθυρο του Inspector του gameobject.

Κάνοντας κλικ στο σκούρο γκρι κουτί που υποδηλώνει την τιμή του JumpCurve, θα εμφανιστεί ένας editor. Σ' αυτόν μπορούμε να δηλώσουμε την καμπύλη του άλματος, δηλαδή τη συνάρτηση στην οποία θα βασιστούμε για να τροποποιούμε το ύψος που βρίσκεται η σφαίρα (τη θέση της στον κατακόρυφο άξονα y). Ας κατασκευάσουμε την καμπύλη χωρίς να μπούμε σε λεπτομέρειες για το πώς δουλεύουν οι καμπύλες Bezier. Θα τις εξετάσουμε σε επόμενο άρθρο.

- Στον editor κάνουμε δεξί κλικ στην τιμή (0,0) κι επιλέγουμε Add Key.
- Το ίδιο κάνουμε και για τις τιμές (0.7,0) και (1,0). Κρατώντας πατημένο το αριστερό κλικ του ποντικιού πάνω σ' ένα κλειδί, μπορούμε να δούμε με μεγάλη ακρίβεια τη θέση του πάνω στους δύο άξονες. Η ροδέλα του ποντικιού ρυθμίζει τη μεγέθυνση των αξόνων. Με το πλήκτρο [Ctrl] και τη ροδέλα, ρυθμίζουμε τη μεγέθυνση μόνο στον άξονα x. Επίσης, με το [Shift] και τη ροδέλα ρυθμίζουμε τη μεγέθυνση μόνο στον άξονα y. Μπορούμε να αλλάξουμε τη θέση ενός κλειδιού σέρνοντάς το.
- Μεταφέρουμε το μεσαίο κλειδί προς τα πάνω, από τη θέση (0.7,0) στη θέση (0.7,0.97).

Το αποτέλεσμα πρέπει να είναι σαν αυτό στην εικόνα που ακολουθεί. Κλείνοντας τον editor η καμπύλη αποθηκεύεται αυτόματα.



Η καμπύλη στην οποία θα βασίσουμε την αλλαγή της θέσης της σφαίρας, στον άξονα y.

Επιστρέφοντας στον κώδικα, παρατηρούμε ότι έχουμε υλοποιήσει και τη μέθοδο Start. Η εν λόγω μέθοδος καλείται αμέσως πριν κληθεί για πρώτη φορά η μέθοδος Update. Ουσιαστικά, η κλήση της υποδηλώνει ότι το αντικείμενο είναι έτοιμο προς rendering. Σε αυτή τη μέθοδο αρχικοποιούμε τη θέση στον άξονα y του αντικειμένου, καθώς και την τιμή που θα μας βοηθήσει στην αναγωγή του χρόνου της καμπύλης. Την τιμή της αναγωγής τη βρίσκουμε διαιρώντας την τιμή του χρόνου που δηλώσαμε στην καμπύλη, με την τιμή του χρόνου της διάρκειας του άλματος. Την τιμή του χρόνου της καμπύλης μπορούμε να την πάρουμε έμμεσα, αναζητώντας την τιμή του χρόνου του τελευταίου key που έχουμε δηλώσει σ' αυτή.

Τέλος, ας κοιτάξουμε και τη μέθοδο handleJump. Σε αυτήν αποθηκεύουμε το χρόνο που έχει περάσει από την αρχή του άλματος, προσθέτοντας το χρόνο που πέρασε για το κάθε καρέ (frame). Με τη βοήθεια της μεθόδου Evaluate, της κλάσης AnimationCurve, παίρνουμε την τιμή της θέσης y που πρέπει να βρίσκεται η σφαίρα (αφού κάνουμε αναγωγή στο χρόνο της καμπύλης) τη δεδομένη χρονική στιγμή. Βασιζόμενοι στο χρόνο που πέρασε, βρίσκουμε την απόσταση που πρέπει να διανύσει η σφαίρα στον άξονα z (για να πάει μπροστά ή πίσω). Κρατώντας τη θέση x της σφαίρας (που έχει ήδη αλλάξει πιο πριν από τη μέθοδο move), τροποποιούμε τις τιμές για τους άξονες y και z του διανύσματος και το θέτουμε ως τη νέα θέση της σφαίρας. Όταν ο χρόνος του άλματος παρέλθει, ενημερώνουμε την κλάση ότι το άλμα τελείωσε θέτοντας τη μεταβλητή jumping σε false.

Τρέχοντας το project μας και πατώντας τα κουμπιά [W] ή [S], βλέπουμε τη σφαίρα να πηδάει εμπρός ή πίσω. Πατώντας ταυτόχρονα τα [A] ή [D] μπορούμε να δούμε ότι η πλευρική κίνηση δεν έχει επηρεαστεί από τις νέες αλλαγές, πράγμα που ήταν κι ο αρχικός σκοπός.

Δοκιμάστε να αλλάξετε τις τιμές των public μεταβλητών ή την καμπύλη του άλματος μέσω του inspector, παρατηρώντας αντίστοιχα τις αλλαγές στη συμπεριφορά της σφαίρας. Οι αλλαγές των τιμών στα components που είναι προσκολλημένα σε ένα game object, μπορούν να αλλάζουν ακόμη κι όταν το παιχνίδι τρέχει. Σ' αυτή την περίπτωση όμως η διακοπή του παιχνιδιού θα επαναφέρει τις αλλαγές στις αρχικές τους τιμές. Σας παροτρύνουμε να υποβάλλετε τις όποιες απορίες έχετε σχετικά με τον κώδικα, αφήνοντας σχόλιο στο site του περιοδικού. Να υπενθυμίσουμε ότι μπορείτε να βρείτε την τελευταία έκδοση του κώδικα στη σελίδα του project στο GitHub και συγκεκριμένα στο https://github.com/ikromm/project-sphere.

Στο επόμενο τεύχος θα προσθέσουμε στο παιχνίδι μας τα πρώτα εμπόδια, τα οποία θα κινούνται προς την ίδια κατεύθυνση ανά γραμμή.


e-card
4000 1234 5678 9010
4000
GOOD THRU
MONTH / YEAR
12 / 19
MasterCard
Cardholder Name
Κάρτα e-coin, που δέχεται Bitcoins: Δουλεύει όπως όλες οι κανονικές κάρτες Debit, παντού στον κόσμο και για online αγορές.
Αποκτήστε τη δική σας με κλικ εδώ
και κερδίστε αυτομάτως
25% έκπτωση
κατά την παραγγελία της κάρτας.